Bernard Odant

Microcontrôleurs 8051 et 8052

Description & mise en œuvre

DUNOD

Bernard ODANT

microcontrôleurs 8051 et 8052

Description et mise en œuvre

DUNOD

Remerciements

J'adresse ici mes remerciements à Mademoiselle Pascale DESMAEL de la Société Intel Corporation, Messieurs Jean-Luc LEVANT de la société MATRA MHS,
Jean-Régis DEMAILLARD, Benoit WEBER de la société SIEMENS,
Dominique PARET de PHILIPS Composants
qui ont mis à ma disposition toute la documentation nécessaire à la rédaction de ce livre.
Je n'oublierai pas de remercier aussi Messieurs Maurice JARAVEL et Francis LAMOTTE de la société RAISONANCE pour leur aide et leurs précieux conseils ainsi que Gilles L'HOTE qui m'a encouragé à écrire ce livre.

Intel est une marque déposée par INTEL Corporation.
MCS est une marque déposée par INTEL Corporation.
MHS est une marque déposée par MATRA-MHS Semiconductor.
Siemens est une marque déposée par SIEMENS AG.
Raisonance est une marque déposée par RAISONANCE SA.

Ce pictogramme mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire le développement massif du **photocopillage**.

Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.

Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation du Centre français d'exploitation du droit de copie (**CFC**, 3 rue Hautefeuille, 75006 Paris).

© Dunod, Paris 1993
ISBN 2-10-001764-0

"Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de l'auteur, ou de ses ayants-droits, ou ayants-cause (loi du 11 mars 1957, alinéa 1er de l'article 40), cette représentation ou reproduction, par quelque procédé que ce soit, constituerai un contrefaçon sanctionnée par les articles 425 et suivants du code pénal. La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinée à une utilisation collective d'une part, et, d'autre part, que les analyses et courtes citations dans un but d'exemple et d'utilisation."

Table des matières

1 - Introduction ____ 1
Les microcontrôleurs MCS 51 ____ 1
Les microcontrôleurs de la famille 8051 ____ 2

2 - Architecture fonctionnelle du noyau 8051 ____ 9
Noyau 8051 ____ 9
L'unité centrale de traitement (U.C.T.) ____ 10
Mémoire programme et mémoire de données ____ 11
Ports d'entrée/sortie (E/S) ____ 11
Les compteurs/temporisateurs (timer 0 et 1) ____ 11
Communication série ____ 12
Les interruptions ____ 12
Circuit d'horloge ____ 12

3 - Présentation matérielle ____ 13
Brochages ____ 13
Notion de fonction secondaire ____ 13
Description des broches ____ 14

4 - Oscillateur, horloge et cycle machine ____ 17
Circuit oscillateur ____ 17
Définition d'un cycle machine ____ 19
Evolution temporelle de l'unité centrale ____ 20
Initialisation du microcontrôleur ____ 21

5 - Mémoire programme et mémoire de données ____ 25
Séparation logique entre programme et données ____ 25
Mémoire programme ____ 26

Microcontrôleur équipé d'une ROM interne ____ 27
Microcontrôleur sans mémoire programme interne ____ 27
Utilisation de l'espace mémoire programme ____ 28
Mémoire de données ____ 29
Mémoire vive interne ____ 29
Accès à des données externes ____ 30
Exemple d'adressage de mémoires externes ____ 32
Mémoire à fonction mixte ____ 32

6 - Modes d'adressage et jeu d'instructions ____ 35
Introduction ____ 35
Modes d'adressages ____ 35
Adressage immédiat ____ 35
Adressage direct ____ 36
Adressage par registre ____ 36
Adressage indirect par registre ____ 36
Adressage indirect indexé par registre ____ 37
Adressage au niveau du bit ____ 38
Classification des instructions ____ 38
Instructions de transfert ____ 38
Instructions arithmétiques ____ 41
Instructions logiques ____ 43
Instructions booléennes ____ 44
Instructions pour rupture de séquence ____ 46
Instructions modifiant les indicateurs ____ 48

7 - Description du jeu d'instructions ____ 49
Syntaxe utilisée ____ 50

8 - Détail de la mémoire vive interne ____ 95
Organisation générale de la RAM interne ____ 95
Registres universels ____ 97
Zone mémoire adressage au niveau du bit ____ 98
Registres à fonction spéciale (SFR) ____ 98
Modes de consommation réduite ____ 102

9 - Conception et utilisation des ports E/S 105
Caractéristiques commmunes aux quatre ports 105
Structure générale d'un port d'entrée/sortie 106
Structure des ports P1, P2 et P3 107
Etage de sortie en technologie HMOS 107
Etage de sortie en technologie CMOS 108
Structure du port P0 109
Ports P0 et P2 comme bus de données et d'adresses 111
Fonctions auxiliaires du port F3 111
Particularité logicielle concernant les quatre ports 112
Chronologie des opérations sur un port 113
Ecriture sur un port (sortie) 113
Lecture d'un port (entrée) 114

10 - Description des timers 115
Généralités 115
TMOD = Registre de contrôle du mode 116
C/T = Choix de la fonction du timer 116
Description du mode opératoire (M1 et M0) 117
Registre de contrôle TCON 120
Timer 2 du 8052 120
Générateur de fréquence de communication série 125

11 - Interface de communication série 131
Présentation 131
Registre de contrôle 132
Fonctionnement en registre à décalage (mode 0) 133
Fonctionnement en mode 1 136
Fonctionnement en mode 2 et 3 139
Communication multiprocesseur 144
Exemples de programmes de gestion du port série 144

12 - Les interruptions 147
Table des vecteurs 148
Exemple d'implantation logicielle 148
Exécution d'une interruption 149

Autorisation et priorité des interruptions ____ 149
Modification de l'ordre de priorité ____ 151
Prise en compte d'une demande d'interruption ____ 152
Interruptions extérieures INT0 et INT1 ____ 153
Interruption par les timers ____ 154
Sources d'interruption timer 0 et 1 ____ 154
Timer 2 du 8052 ____ 154
Port série ____ 155
Temps de réponse ____ 155
Exemple d'utilisation ____ 156
Exécution d'un programme en mode pas à pas ____ 156
Gestion du port série sous interruption ____ 157

13 - Processeur booléen ____ 165
Définition du terme "processeur booléen" ____ 165
Intérêt de l'utilisation du processeur booléen ____ 166
Programmation d'un circuit de logique combinatoire ____ 167
Solution utilisant un adressage au niveau de l'octet ____ 169
Solution utilisant les instructions de test de bit ____ 170
Utilisation des instructions booléennes ____ 170
Programmation d'un circuit de logique séquentielle ____ 172
Programmation d'une séquence simple ____ 173
Divergence et convergence en ET ____ 177
Divergence et convergence en OU ____ 179

14 - Aide au développement matériel et logiciel ____ 181
Elaboration d'un projet ____ 181
Choix de l'organisation matérielle ____ 182
Programme interne ou externe ? ____ 182
Organisation de la RAM interne ____ 184
Composants de la famille MCS-80/85 ____ 185
Extension de la mémoire programme à plus de 64 ko ____ 186
Développement du logiciel ____ 189
Fichier SOURCE ____ 189

Outils de développement ____ 192
Simulateur de programmes ____ 192
Simulateur d'EPROM ____ 193
Emulateurs ____ 193

15 - Ressources logicielles ____ 197
Module arithmétique ____ 197
Multiplication 16 bits par 16 bits ____ 200
Division 32 bits par 16 bits ____ 202
Division 16 bits par 16 bits ____ 205
Conversion en format BCD ____ 207
Division par 256 ____ 208
Gestion d'un afficheur L.C.D. ____ 208

Annexe = Précis de programmation ____ 211

1

Introduction

Du Minitel aux terminaux électroniques de paiement, des automates programmables industriels aux variateurs de vitesse pour moteurs électriques, le microcontrôleur 8051 ou l'un des membres de sa famille a su participer à la réalisation de ces projets.

Cet ouvrage propose l'étude du 8051 et de sa première déclinaison, le 8052. Cette étude permettra au lecteur de choisir et d'utiliser l'un des nombreux membres d'une famille de microcontrôleurs dont l'origine et l'importance sont précisées dans ce chapitre.

Les microcontrôleurs MCS 51

Au début des années 80, la société INTEL proposa, comme une suite logique au microcontrôleur 8048, un nouveau modèle baptisé de la référence 8051. Cette première version réalisée en technologie HMOS, relativement gourmande en énergie, se déclinait déjà selon trois versions, le 8051 proprement dit doté d'une ROM interne programmée à la fabrication, le 8031 dépourvu de mémoire morte et le 8751 équipé d'une EPROM. Cette première série se compléta rapidement de la version 8052. Ce dernier microcontrôleur très proche du 8051, est enrichi d'un périphérique (un troisième timer) et de 128 octets de RAM interne. Une version du 8052 est proposée avec un langage BASIC implanté dans sa mémoire ROM.

La famille MCS-51 prend ainsi naissance et se complète rapidement d'autres membres. La multiplicité des modèles est justifiable par le concept déjà évoqué : l'intégration d'un environnement minimal autour d'une unité centrale de traitement limite l'universalité du composant ; il faut donc envisager de reproduire la même unité centrale en modifiant seulement l'environnement intégré. L'intérêt de cette démarche est bien sûr en premier lieu, l'assurance pour l'utilisateur, de trouver le microcontrôleur le mieux adapté à son projet.

A cet avantage il faut ajouter le fait que tous ces microcontrôleurs étant conçus autour d'un même noyau matériel et logiciel, l'investissement engagé dans l'étude et l'emploi d'un des microcontrôleurs est donc réutilisable sur les autres membres de cette même famille. La force d'une famille de microcontrôleurs réside donc dans sa capacité à se diversifier.

Cette perspective incita un bon nombre de constructeurs (SIEMENS, MHS, OKI, SIGNETICS, PHILIPS, etc.) à passer des accords de production en seconde source de ces microcontrôleurs. L'acquisition d'une licence de production du noyau de base constituant le 8051 autorise alors ces sociétés à concevoir à leur tour de nouvelles versions originales.

Concevoir un nouveau microcontrôleur appartenant à la famille 8051 consiste à compléter le masque de fabrication du 8051 par de nouvelles fonctions périphériques.

L'unité de la famille est donc assurée par le fait que toutes les nouvelles versions sont construites autour d'un même noyau logiciel et matériel. La diversité existe au niveau du choix des fonctions périphériques intégrées autour ce de noyau.

Il n'est plus possible alors de parler seulement de famille MCS-51 propre au constructeur INTEL.

Les microcontrôleurs de la famille 8051

Cette appellation recouvre tous les microcontrôleurs conçus autour du noyau matériel et logiciel que constitue le masque de fabrication du 8051. Cette désignation étant alors partagée par plusieurs constructeurs.

L'importance du nombre des versions peut troubler l'utilisateur lorsqu'il est question d'établir un choix. Il est alors nécessaire de consulter les catalogues des différents fabricants. Quatre tableaux sont proposés dans les pages suivantes. Ces tableaux présentent seulement la production de quatre constructeurs. La liste n'est donc pas complète mais elle permet quand même d'entrevoir la richesse de la famille 8051.

La présentation sous forme de tableaux donne une bonne vue d'ensemble des principales caractéristiques des différentes versions, mais il est évident qu'une telle présentation ne peut permettre de préciser toutes les nuances existant entre les modèles proposés. Les références données peuvent se compléter de suffixes différenciant par exemple :

— la fréquence maximale de l'oscillateur ;
— les limites de température pour lesquelles le fonctionnement est garanti ;
— les types de boîtiers proposés pour chaque version ;
etc.

L'utilisation d'abréviations dans ces tableaux nécessite quelques précisions.

WDT provient de Watch Dog Timer que l'on traduit par chien de garde. Matériellement la fonction chien de garde est réalisée par un compteur 16 bits. Certaines versions intègrent un convertisseur analogique numérique CA/N. Ce convertisseur se caractérise par une résolution exprimée en bit et un nombre d'entrées qui peut être supérieur à l'unité (utilisation d'un multiplexeur temporel). Il est donc précisé le nombre d'entrées et la résolution du CA/N.

La même présentation est retenue pour le nombre des sources d'interruption et le nombre de niveaux de priorité qui peuvent leur être attribuées. Ainsi 5/2 signifie cinq sources d'interruption sur deux niveaux de priorité.

Les compteurs/temporisateurs peuvent permettre d'obtenir un signal de sortie modulé en largeur d'impulsion (PWM) ou de servir en entrée de capture (CAP).

Dans les chapitres suivants, l'appellation unique de 8051 sera utilisée pour la description des différentes versions 8051, 8031 et 8751. De même, le générique 8052 sera utilisé pour préciser les différences présentées par les versions 8052, 8032 et 8752 par rapport au noyau 8051. Les descriptions faites dans ce livre ne concernent donc que les versions de base de la famille ou d'une façon plus générale la base commune à tous les microcontrôleurs élaborés autour du noyau 8051.

VERSION			MEMOIRE		Timer	PORT	Interrup.	PORT	Fréquence	Particularités
sans ROM	ROM	EPROM	ROM	RAM	16 bits	8 bits	/ niveau	série	en MHz	
8031	8051		4K	128	2	4	6/2	UART	3.5 à 12	version N MOS
8031AH	8051AH	8751H	4Ko	128	2	4	6/2	UART	3.5 à 12	
		8751H8	"	"	"	"	"	"	3.5 à 8	
		8751BH	"	"	"	"	"	"	"	
8032AH	8052AH	8752BH	8Ko	256	3	4	8/2	UART	3.5 à 12	
80C31BH	80C51BH	87C51	4Ko	128	2	4	6/2	UART		CMOS avec modes de consommation réduite
80C51FA	83C51FA	87C51FA	8Ko	256	3+PCA	4	7/2	UART		possibilité de sorties PWM, entrées de capture et chien de garde
–	83C51FB	87C51FB	16Ko	256	3+PCA	4	7/2	UART		
80C51GA	83C51GA	87C51GA	4Ko	128	2	4	7/2	2		8 entrées analogiques 8 bits, WDT et port série synchrone
80C152JA	83C152JA		8Ko	256	2	5	11/2	2		port série multi protocole et 2 canaux de DMA
80C152JB			0	256	2	7	11/2	2	3.5 à 12/16	
80C152JC	83152JC	"	8Ko	256	2	5	11/2	2		
80C152JD			0	256	2	7	11/2	2		
80C451	83C451		4Ko	128	2	7	5/2	UART		7 ports
80C452	83C452	87C452P	8Ko	256	2	5	8/2	UART	3.5 à 14	2 canaux DMA

Tableau 1.1 : Gamme des microcontrôleurs MCS—51 INTEL (1991)

VERSION			MEMOIRE		Timer	PORT	Interrup.	PORT	Fréquence	Particularités
sans ROM	ROM	EPROM	ROM	RAM	16 bits	8 bits	/ niveau	série	en MHz	
8031	8051		4Ko	128'	2	4	5/2	1	12,16,20	
8032	8052		8Ko	256	3	4	6/2	1	12,16,20	
	80513		16Ko	256	3	4	6/2	1	12,16	
80535	80515		8Ko	256	3 +WDT	6	12/4	1	1.2 à 12	8 entrées analogiques CAN 8 bits; 4 sorties PWM; 4 entrées capture; 4 niveaux de priorité sur interruption
	80515-4		16Ko	256	3 +WDT	6	12/4	1	1.2 à 12	
80C32	80C52		8Ko	256	3	4	6/2	1	12,16,20	
80C535	80C515		8Ko	256	3 +WDT	6	12/4	1	0.5 à 16	Version CMOS des 80515/80535
80C537	80C517		8Ko	256	3 +WDT	7	14/4	2	1 à 16	12 entrées analogiques; 8 pointeurs 16 bits; unité arithmétique 16 bits; 21 sorties PWM; 5 entrées "capture" 2 ports série; 4 TIMERS 16 bits WDT programmable
80C515A	83C515A 5		32Ko	1.25 K	3 + 2 WDT	6	12/4	1	0.5 à 18	CA/N de résolution 10 bits; RAM interne augmentée de 1Ko.
80C517A	83C517A 5		32Ko	2.25 K	4 + 2 WDT	7	14/4	2	1 à 18	RAM interne de 2.25 Ko. CA/N 10 bits

Tableau 1.2 : Gamme des microcontrôleurs SIEMENS (1991)

VERSION			MEMOIRE		Compteur	PORT	PORT	Fréquence	Particularités
sans ROM	ROM	EPROM	ROM	RAM	/tempo.	8 bits	série	en MHz	
–	83C751	87C751	2K	64	1(16 BITS)	2+3/8	I^2C	12,16	
–	83C752	87C752	2K.	64	1	2+5/8	I^2C	12 & 16	PWM & CA/N 5E/8 bits
8031AH	8051AH	–	4K.	128	2	4	UART	12	NMOS
80C31B	80C51B	87C51	4K.	128	2	4	UART	12,16,24,33	CMOS
80CL31	80CL51	–	4K.	128	2	4	UART	0.032 à 16	Vcc de 1.8 à 6 V.
80CL410	83CL410	–	4K.	128	2	4	I^2C	0.032 À 16	VCC DE 1.8 À 6 V.
80C451	83C451	87C451	4K.	128	2	7	UART	12&16	
80C550	83C550	87C550	4K.	128	2 + WDT	4	UART	12 & 16	CA/N 8E/8 bits
80C851	83C851	–	4K.	128	2	4	UART	12	EEPROM de 256 octets
–	83C852	–	6K.	256	2	2/8		12	implanté sur carte
8032AH	8052AH	–	8K.	256	3	4	UART	12 ou 15	NMOS
80C32	80C52	87C52	8K.	256	3	4	UART	16, 20 & 24	CMOS
80C575	83C575	87C575	8K.	256	3+PCA +WDT	4	UART amélioré	16, 20 & 24	4 comparateurs analo-giques et reset sur "0"
80C552	83C552	87C552	8K.	256	3+ WDT	6	UART+ I^2C	16, 20 & 24	2 sorties PWM & CA/N 8E/10 bits
80C562	83C562	–	8K.	256	3+ WDT	6	UART	16	2 sorties PWM & CA/N 8E/10 bits
80C652	83C652	87C652	8K.	256	2	4	UART+ I^2C	16, 20 & 24	même brochage que 8051
–	83C053	–	8K.	192	2	3 4/8	–	12	affichage vidéo et 9 sorties PWM
–	83C054	87C054	16K.	192	2	3 4/8	–	12	idem
–	83C654	87C654	16K.	256	2	4	UART+ I^2C	16, 20 & 24	
–	83CE654	–	16K.	256	2	4	UART+ I^2C	16, 20 & 24	
80C592	83C592	87C592	16K.	512	3+ WDT	6	UART+ CAN	16	CA/N 8E/10 bits, 2 sorties PWM & interface bus CAN
80C528	83C528	87C528	32K.	512	3+ WDT	4	UART+ I2C	16, 20 & 24	

Tableau 1.3 : Gamme des microcontrôleurs PHILIPS (1992)

VERSION		MEMOIRE		Timer	PORT	Interrup.	PORT	Fréquence	Particularités
sans ROM	avec ROM	ROM	RAM	16 bits	8 bits	/ niveau	série	en MHz	
80C31 80C31-1 80C31-S	80C51 80C51-1 80C51-S	4Ko	128	2	4	5/2	1	de 0 à 12 de 0 à 16 de 0 à 20	versions CMOS totalement statiques dont la fréquence d'horloge peut être réduite jusqu'à 0 sans perte de données des registres ou de la RAM interne
80C32	80C52	8Ko	256	3	4	6/2	1	de 0 à 16 ou 0 à 16 ou 0 à 20	
80C154	83C154	16Ko	256	3	4	6/2	1	de 0 à 12 ou 0 à 16	Impédance programmable des ports, timer 0 et 1 pouvant former un timer de 32 bits ou un compteur chien de garde Le suffixe -L désigne des produits pour lesquels Vcc peut être compris entre 2.7V et 5.5V
80C154-L	83C154-L 83C154D 83C154D-L	32Ko	256	3	4	6/2	1	de 0 à 6 de 0 à 12 ou 0 à 16 de 0 à 6	
80C31µ	80C51µ	4Ko	128	2	4	5/2	1	de 0 à 12, 16, 20, 25, et 30 MHz	Nouvelles versions réalisées en technologie CMOS rapide (SCMOS). Il est possible d'obtenir des versions pouvant fonctionner jusqu'à 42 MHz.
80C32µ	80C52µ	8Ko	256	3	4	6/2	1	de 0 à 12 16, 20, 25, et 30 MHz	
80C154µ	83C154µ	16Ko	256	3	4	6/2	1	de 0 à 20, 25, 30 MHz	

Tableau 1.4 : Gamme des microcontrôleurs MHS electronic (1992)

2

Architecture fonctionnelle du noyau 8051

Noyau 8051

Le microcontrôleur 8051 est le noyau à partir duquel ont été conçus les autres membres de la famille MCS-51 d'INTEL et les dérivés des autres constructeurs.

Ce chapitre ne constitue qu'un rapide descriptif des éléments constitutifs et de leurs principales caractéristiques. La figure 2.1 donne le schéma bloc de ce noyau alors que la figure 2.2 précise les différences apportées aux versions 8052.

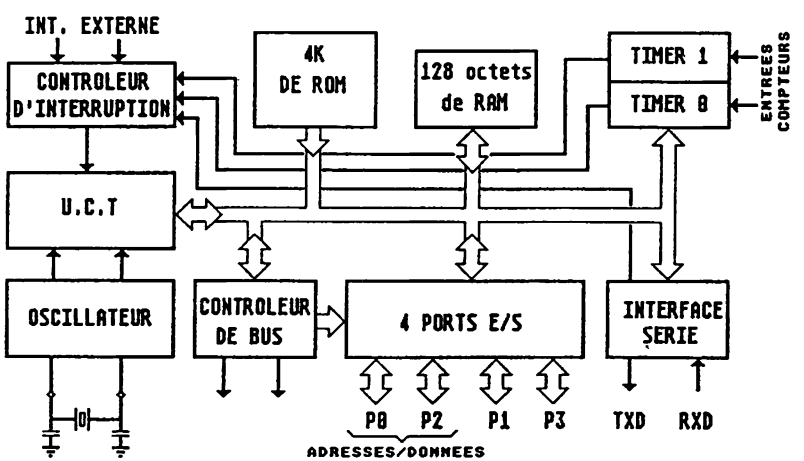


Figure 2.1 : Schéma bloc du noyau 8051

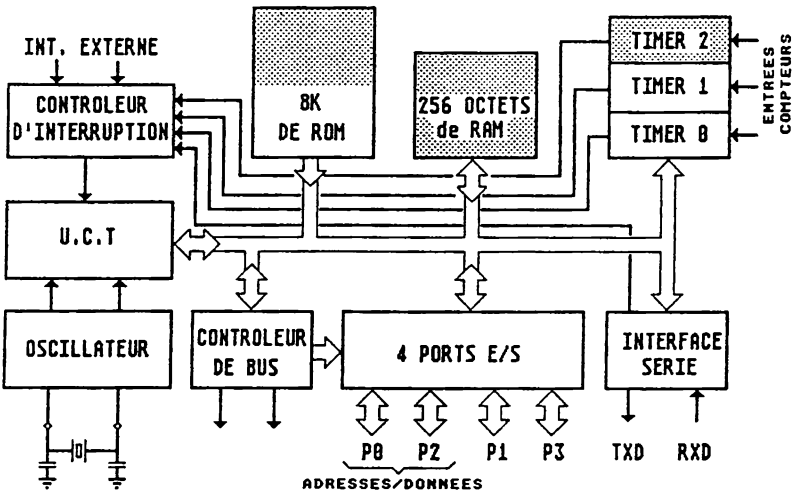


Figure 2.2 : Schéma bloc de la version 8052

L'unité centrale de traitement (U.C.T.)

Ce sous-ensemble comporte en fait plusieurs éléments constitutifs d'un microprocesseur. L'absence de détails sur la structure interne de l'unité centrale peut se comprendre de part la conception très proche de celle d'un microprocesseur 8 bits classique. On peut imaginer qu'elle comporte entre autre :

— une unité de contrôle capable de récupérer un code instruction, de le décoder et de lancer une séquence propre à son exécution. La plupart de ces séquences d'exécution ne durent qu'un cycle machine, c'est-à-dire douze périodes d'horloge interne.

— une unité arithmétique et logique. Pour cette partie, l'étude du jeu d'instructions permet de mieux évaluer ces caractéristiques. Avant de procéder à cette étude détaillée on peut simplement préciser que des multiplications et des divisions sur des formats de huit bits sont possibles. L'unité centrale de traitement est optimisée pour des applications de contrôle de processus.

Il est possible d'utiliser cette unité centrale 8 bits comme une unité de traitement au niveau du bit ce qui facilite les opérations logiques fréquentes : prise en compte de l'état logique d'une entrée (capteur), modification de l'état d'une sortie (voyant, relais, transistor, etc.). Cette possibilité fait employer au constructeur la terminologie de

processeur booléen. Le microcontrôleur 8051 offre donc les possibilités et la puissance d'un processeur 8 bits, complétées par la souplesse d'un processeur du type monobit parfaitement adapté aux problèmes d'automatismes

Mémoire programme et mémoire de données

Le 8051 autorise un adressage de 64 Koctets de mémoire programme et de 64 Koctets pour la mémoire de données. En plus il offre une mémoire vive interne de 128 octets. Cette notion d'adressage distinct entre programme et données permet de doubler l'espace de mémoire adressable. La présence d'une mémoire morte de 4Ko. intégrée faisant partie de la mémoire programme et d'une mémoire vive interne de 128 octets peut sembler contredire cette possibilité. Il faut en effet, noter en plus de la distinction entre mémoire programme et mémoire de données, une autre distinction possible : mémoire interne et mémoire externe. L'adressage de mémoires externes restant une option à la vocation première du microcontrôleur : se suffire à lui seul. De ce fait, placer des composants mémoires externes reste une opération possible, mais il faut alors trouver quelque part les bus de données et d'adresses. Les deux ports parallèles de 8 bits P0 et P2 perdent alors leur fonction de port et jouent le rôle de bus externes. Si les ports P0 et P2 ont cette fonction auxiliaire, il faudra aussi amputer le port P3 de deux de ces entrées-sorties pour assurer le contrôle de ces bus d'adresses et de données externes.

Ports d'entrée/sortie (E/S)

32 lignes d'entrée/sortie individuellement adressables sont réparties en 4 ports, désignés par les abréviations P0, P1, P2 et P3. Chaque ligne est définie par le constructeur comme quasi bidirectionnelle, c'est-à-dire pratiquement utilisable comme une entrée ou une sortie. En plus de cette fonction principale, certaines des broches du circuit intégré correspondant à ces lignes d'entrée/sortie peuvent remplir une fonction dite auxiliaire.

Les compteurs/temporisateurs (timer 0 et 1)

Le mot anglo-saxon timer représente un sous-ensemble capable d'assurer des fonctions comme le comptage ou la temporisation. Le terme timer sera conservé dans la suite du texte en remplacement du terme compteur/temporisateur.

Le 8051 possède deux timers de 16 bits, le 8052 en a trois. En mode temporisateur, l'incrémentation des registres a lieu une fois par cycle machine. La fréquence maximale de comptage est donc égale au 1/12 de la fréquence d'horloge du processeur. En mode compteur, l'incrément du registre a lieu à chaque flanc descendant enregistré sur l'entrée associée au compteur. La fréquence maximale de comptage est limitée alors

à une fréquence 24 fois plus faible que celle de l'horloge du processeur. Les timers 0 et 1 connaissent quatre modes de fonctionnement différents permettant de modifier le format du registre de comptage ou d'autoriser un rechargement automatique à partir d'une valeur de consigne.

Le timer 1 peut faire office de générateur de fréquences pour les transmissions par le port série.

Le timer 2 connaît trois modes de fonctionnement particuliers :

— comptage sur 16 bits avec rechargement automatique,
— comptage sur 16 bits avec sauvegarde automatique,
— générateur de fréquences de transmission.

Communication série

Le port série bidirectionnel est capable d'émettre et de recevoir simultanément. La réception d'une donnée peut commencer alors que la donnée précédente n'a pas encore été prélevée dans le registre de réception. Ce port série possède quatre modes de fonctionnement qui diffèrent par le format de la donnée et le débit de transmission. Ces vitesses de transmission sont programmables à partir du timer 1.

Les interruptions

Sur un 8051 il y a cinq sources d'interruption possibles et six sur le 8052. Ce sont deux entrées que l'on programme à sa convenance pour qu'elles réagissent soit à des niveaux logiques, soit à des flancs de transition, les compteurs/temporisateurs 0 et 1 (et 2 pour le 8052) et enfin le port série en réception ou en émission.

Par programmation il est possible de fixer deux niveaux de priorité d'interruption : si une interruption à niveau prioritaire élevé intervient pendant le traitement d'un interruption à niveau inférieur, le processeur honore aussitôt la demande à priorité élevée. Chacune des sources d'interruption possède un vecteur dont se sert le processeur afin d'exécuter le sous-programme convenable après sauvegarde du PC. Ces vecteurs correspondent à des adresses fixes situées en partie basse de la mémoire programme.

Circuit d'horloge

Un circuit oscillant est intégré dans le noyau. Les seuls éléments extérieurs à prévoir sont principalement un quartz et deux condensateurs. Le choix du quartz permet à l'utilisateur de définir la vitesse d'exécution des instructions.

3

Présentation matérielle

Brochages

Les microcontrôleurs 8051, 8052, et leur version CMOS sont disponibles chez tous les constructeurs en boîtier DIP 40 broches ou en boîtier PLCC 44 broches. Des constructeurs proposent ces microcontrôleurs sous d'autres boîtiers. Comme l'indique la figure 3.1, les versions 8052 se distinguent par deux fonctions secondaires pour les broches P1.0 et P1.1. Il s'agit de T2 et T2EX qui représentent les entrées du troisième compteur/temporisateur.

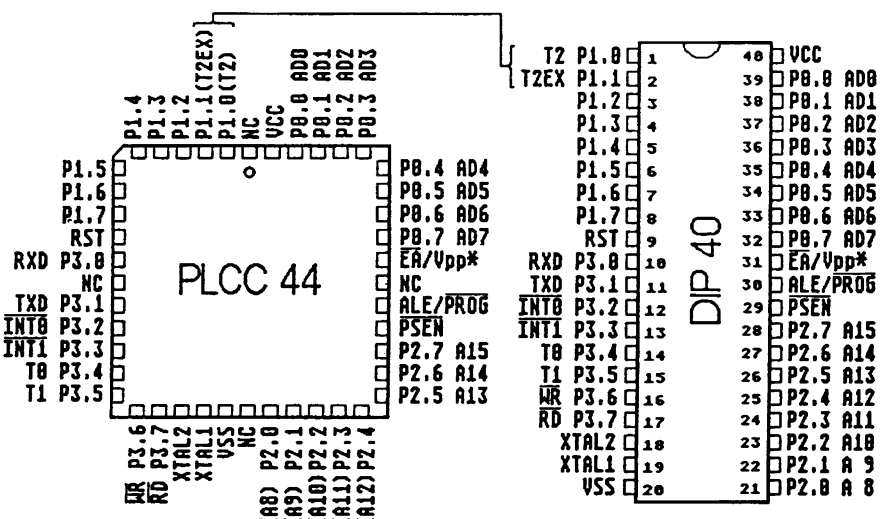


Figure 3.1 : Brochage des 8051 et 8052

Notion de fonction secondaire

Les quatre ports P0, P1, P2 et P3 constituent un ensemble de 32 lignes bidirectionnelles pouvant être utilisées aussi bien en entrées qu'en sorties indépendantes les unes des autres et ce, sans procédure d'initialisation particulière. Chacune

de ces lignes bidirectionnelles est matérialisée par une broche du circuit intégré. Pour les microcontrôleurs présentés en version DIP 40 broches il ne reste donc que huit broches pour assurer les autres interconnexions à l'environnement !

Si cette profusion de lignes d'entrée/sortie correspond bien aux besoins d'une commande d'un processus simple, elle limite du même coup les possibilités d'extension du système, à moins d'augmenter considérablement la taille et le nombre de connexions du circuit intégré pour que celui-ci puisse gérer en plus de ces quatre ports un bus de données et un bus d'adresses.

Sans ces bus externes il y a automatiquement une limitation à l'adaptabilité du produit. Bien que la fonction première d'un microcontrôleur soit de permettre un contrôle et une commande d'un système limité et fini, il faut quand même qu'il puisse s'adapter tant bien que mal à des contextes plus larges et offrir ainsi une alternative à l'utilisation d'un microprocesseur.

Une solution consiste alors à doter certaines broches correspondant à des lignes d'entrée/sortie, d'une deuxième fonction ou fonction auxiliaire. C'est le cas des broches correspondant aux ports P0 et P2 qui pourront devenir bus de données et d'adresses, des broches du port P3 qui pourront devenir des signaux de contrôle des bus externes ou assurer des fonctions diverses telles que transmission et réception en mode série ou source d'interruption externe.

Description des broches

V_{CC} et V_{SS} :

Broches d'alimentation. La broche V_{CC} doit recevoir une tension de +5 V à 10 % prés, par rapport à la broche V_{SS}.

RST :

Entrée d'initialisation. Un niveau logique 1 sur cette entrée pendant une durée correspondant à deux cycles machine, provoque une initialisation du microcontrôleur. Pour que l'initialisation soit effective, il faut que l'horloge soit opérationnelle durant ces deux cycles machine. La figure 4.9 du chapitre 4 présente une structure minimum assurant une bonne initialisation du microcontrôleur lors de sa mise sous tension.

$\overline{EA}$ (External Access) :

Cette entrée, active au niveau logique 0, permet de configurer le microcontrôleur en système avec bus externes. Si après initialisation la borne $\overline{EA}$ est reconnue comme étant reliée à la masse ou à un état logique 0, les ports P0 et P2 remplissent alors les fonctions de bus d'adresses et de données, les instructions du programme sont

recherchées alors à partir de l'adresse 0000 dans le plan mémoire externe. Si $\overline{EA}$ est reliée au potentiel V_{CC} ou à l'état logique 1, le code programme est recherché dans la mémoire interne. Les versions du microcontrôleur dépourvues de mémoire programme interne doivent donc obligatoirement être utilisées avec $\overline{EA}$ reliée au potentiel V_{SS}. Pour les versions du microcontrôleur dont la mémoire programme interne est une EPROM, cette entrée recevra la tension (V_{PP}) de programmation.

$\overline{PSEN}$ (Programm Store ENable) :

Le fait que la famille 8051 puisse adresser 64Ko. de mémoire programme et 64 Ko. de mémoire de données implique un bus d'adresses de 17 bits. La broche $\overline{PSEN}$ joue, en quelque sorte, le rôle du dix-septième bit. $\overline{PSEN}$ passe à l'état logique 0 dès que le microcontrôleur entreprend la récupération d'une instruction dans la mémoire programme externe. Cette sortie n'est donc active que si $\overline{EA} = 0$ et doit être utilisée comme signal de sélection des circuits mémoires ROM ou EPROM. Il faut noter que lors d'un accès à la mémoire externe de données, cette sortie reste à l'état 1. Elle passe deux fois à l'état 0 durant un cycle machine correspondant à l'accès de la mémoire programme externe.

ALE (Adress Latch Enable) :

La broche ALE est une sortie prévue pour commander le démultiplexage du port P0 lorsque celui-ci est validé dans sa fonction secondaire : partie basse du bus d'adresses et bus de données.

lorsque ALE est dans l'état 1, le port P0 présente la partie A0/A7 de l'adresse. Lors de la transition 1 vers 0 de ALE, l'adresse toujours présente doit être mémorisée grâce à un circuit externe. Durant la période où ALE = 0 le port P0 devient alors bus de données.

Cette broche est aussi utilisée comme une entrée lors de la phase de programmation de l'EPROM interne des versions 8751 ou 8752.

XTAL1 et XTAL2 :

XTAL1 est l'entrée de l'amplificateur inverseur destiné à l'oscillateur d'horloge, alors que XTAL2 représente sa sortie. Les éléments externes à rajouter pour compléter le circuit d'horloge sont un quartz à placer entre XTAL1 et XTAL2 et 2 condensateurs reliant ces 2 broches au potentiel V_{SS}. Il reste cependant la possibilité de cadencer le microcontrôleur par un circuit d'horloge externe.

P0.0 à P0.7 (port P0) :

Les 8 lignes du port P0 sont du type "à drain ouvert". Cette particularité est justifiée par la fonction auxiliaire attribuée à ce port : bus d'adresses et de données lorsque

$\overline{EA}$= 0. Dans ce mode d'utilisation chaque ligne de bus peut supporter l'équivalent de 8 charges TTL LS. Lorsque le port P0 est utilisé en tant que sortie, il faut alors l'équiper d'un réseau de résistances de rappel au potentiel $+V_{CC}$. Une description plus complète de ce port ainsi que des trois autres fait l'objet du chapitre 9.

P1.0 à P1.7 (port P1) :

P1 est un port bidirectionnel de huit bits. L'étage de sortie (équipé d'une résistance interne de rappel au +5 V) peut piloter des courants équivalents à quatre charges TTL LS.

P2.0 à P2.7 (port P2) :

P2 est un port bidirectionnel de huit bits. L'étage de sortie (équipé d'une résistance interne de rappel au +5 V) peut piloter des courants équivalents à quatre charges TTL LS.

La fonction secondaire de ce port est de fournir la partie haute de l'adresse (A8 à A15) lors d'un accès à la mémoire externe. Certains modèles comme le 8051AHP, pour lequel la ROM interne est protégée, l'espace d'adressage externe est limité à 4 Ko. Pour cette version les bits P2.4 à P2.7 restent forcés à 0.

P3.0 à P3.7 (port P3) :

Le port P3 possède des caractéristiques de sortie et d'entrée similaires aux ports P1 et P2. Les fonctions secondaires attribuées aux lignes de ce port sont :

pour

P3.0	RXD	entrée de l'interface série ;
P3.1	TXD	sortie de l'interface série ;
P3.2	$\overline{INT0}$	entrée pour interruption externe ;
P3.3	$\overline{INT1}$	entrée pour interruption externe ;
P3.4	T0	entrée de comptage pour timer 0 ;
P3.5	T1	entrée de comptage pour timer 1 ;
P3.6	$\overline{WR}$	sortie écriture de la mémoire externe ;
P3.7	$\overline{RD}$	sortie lecture de la mémoire externe.

Ces deux derniers signaux sont actifs sur état 0.

Remarque : A chacun de ces ports est associé un registre interne situé dans la mémoire vive interne. Ces registres portent le même nom que le port auquel ils sont associés. Lorsqu'une ligne de port doit être utilisée comme entrée, il est impératif de maintenir le bit du registre qui lui correspond dans l'état logique 1, état dans lequel il est placé d'ailleurs après initialisation. Cette remarque est définie dans le chapitre 9.

4

Oscillateur, horloge et cycle machine

Les broches XTAL1 et XTAL2 représentent respectivement l'entrée et la sortie d'un étage inverseur qui peut être transformé en oscillateur du type PIERCE par la mise en place d'un quartz et de deux condensateurs. L'oscillateur ainsi réalisé commande le circuit d'horloge interne du microcontrôleur. La fréquence du signal d'horloge interne est égale à la moitié de la fréquence de l'oscillateur (fréquence du quartz). L'horloge interne permet de définir des phases internes P, des états internes S et la notion de cycle machine.

Circuit oscillateur

La réalisation de l'oscillateur est représentée en figure 4.1.

Elle utilise un circuit interne différent selon la technologie utilisée pour la fabrication du microcontrôleur.

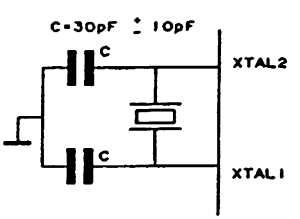


Figure 4.1 : Circuit oscillateur

Comme le montre la figure 4.2, pour les microcontrôleurs réalisés en technologie HMOS la connexion vers le séquenceur interne est réalisée au niveau de la broche XTAL2 alors que dans les versions CMOS (figure 4.3) cette connexion est réalisée du côté de XTAL1. Du point de vue de l'utilisateur ces modifications de structure interne ne modifient en rien une utilisation directe du circuit oscillateur et le montage de la figure 4.1 reste valable.

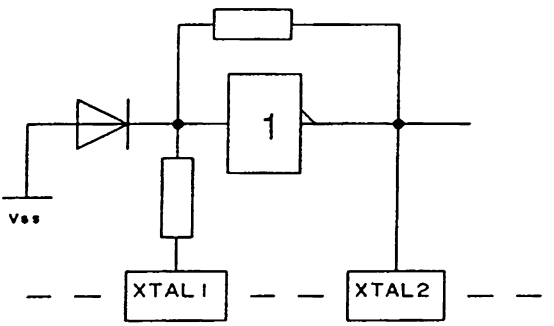


Figure 4.2 : Structure interne de l'oscillateur version HMOS

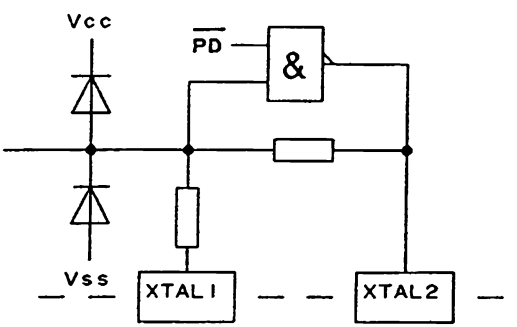


Figure 4.3 : Structure interne de l'oscillateur (version CMOS)

Par contre, dans le cas où le concepteur voudrait commander le microcontrôleur par un circuit externe, ces considérations seront à prendre en compte :

Pour les modèles réalisés en technologie HMOS, le signal externe doit être amené sur XTAL2 alors que XTAL1 est connectée au même potentiel Vss.

Pour les modèles réalisés en technologie CMOS, le signal externe est relié à XTAL1 et la broche XTAL2 reste non connectée. Les figures 4.4 et 4.5 résument cette description.

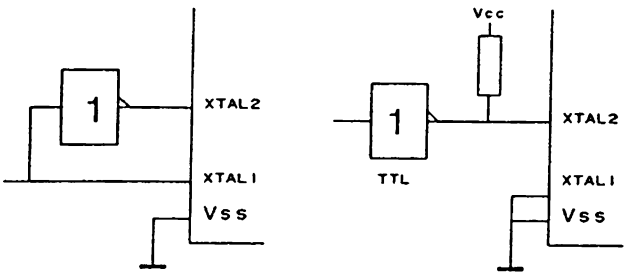


Figure 4.4 : Câblage d'une horloge externe pour les versions HMOS (NMOS)

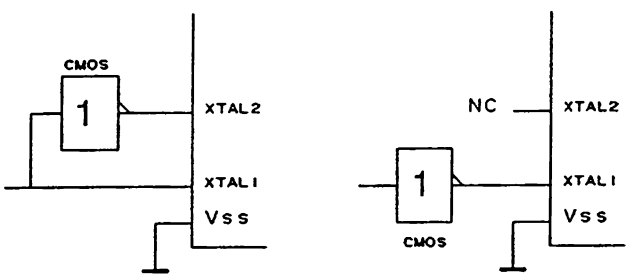


Figure 4.5 : Câblage d'une horloge pour les versions CMOS

Définition d'un cycle machine

L'oscillateur ainsi réalisé commande un circuit interne d'horloge. Ce circuit d'horloge définit une séquence d'états repérés de S1 à S6. Chaque état S se décompose en deux phases P1 et P2. Un état correspondant à une durée de deux périodes du signal issu de l'oscillateur, on remarque donc qu'une phase correspond à une période de la fréquence de l'oscillateur. Les appellations utilisées dans les chapitres suivants utilisent donc ces notions d'états et de phases dans l'état. Pour exemple, le repère S5P2 représente donc la période état 5 phase 2.

Comme le fait apparaître la figure 4.6, un cycle machine est constitué de 6 états (de S1 à S6) soit 12 périodes de l'oscillateur. Pour un quartz de 12 MHz, la durée d'un cycle machine est donc d'une microseconde.

Un cycle machine commence sur la période S1P1 et se termine sur la période S6P2.

Évolution temporelle de l'unité centrale

La figure 4.6 représente les chronogrammes de recherches et d'exécutions d'instructions.

Les lignes A et B représentent l'exécution d'instructions exécutables en un seul cycle machine. Le code opération est lu sur la période S1 et pendant que ce code est mémorisé dans le registre instruction, un deuxième cycle de récupération de code est effectué durant la période S4. Cette deuxième récupération n'est prise en compte qu'en fonction du type d'instruction repéré sur le décodage du premier code. Dans tous les cas l'exécution complète s'achève avec la période S6.

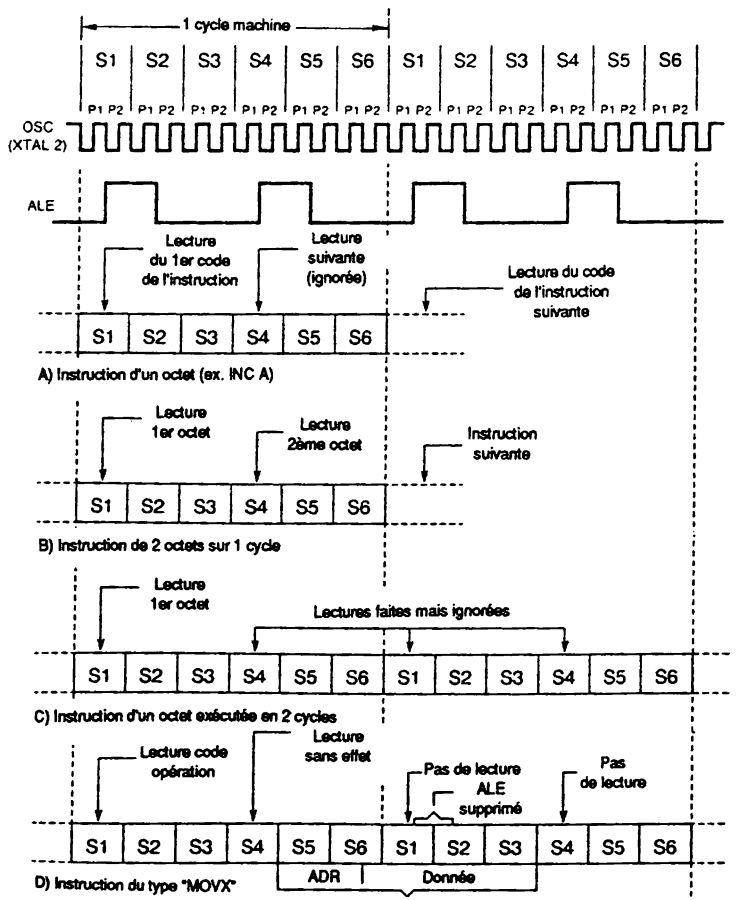

1 cycle machine
S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6
P1 P2
OSC (XTAL 2)
ALE
Lecture du 1er code de l'instruction
Lecture suivante (ignorée)
Lecture du code de l'instruction suivante
A) Instruction d'un octet (ex. INC A)
Lecture 1er octet
Lecture 2ème octet
Instruction suivante
B) Instruction de 2 octets sur 1 cycle
Lectures faites mais ignorées
C) Instruction d'un octet exécutée en 2 cycles
Lecture code opération
Lecture sans effet
Pas de lecture
ALE supprimé
Pas de lecture
D) Instruction du type "MOVX"
ADR
Donnée

Figure 4.6 : Récupérations et exécutions d'instructions

La ligne C décrit le déroulement de l'instruction INC DPTR qui signifie incrémenter le pointeur 16 bits DPTR. Ce pointeur est en réalité composé de deux parties de 8 bits. L'instruction est codée sur un seul octet, mais son exécution nécessite deux cycles machines. La récupération de code est donc inhibée sur la périodes S4 du premier cycle machine et sur les périodes S1 et S4 du deuxième cycle.

L'instruction MOVX représente un transfert entre accumulateur et mémoire externe de données. La ligne D représente la chronologie d'une telle instruction qui se déroule en deux cycles machine. Aucun cycle de récupération n'est produit durant le second cycle machine. C'est le seul cas où des cycles de récupération sont supprimés.

Les cycles de récupération et d'exécution s'effectuent de la même façon et dans le même temps pour un programme situé en mémoire externe que pour un programme situé en mémoire interne.

Une récupération d'instructions depuis un programme externe est illustrée par la figure 4.7(a). Cette figure montre la chronologie des signaux ALE et $\overline{\text{PSEN}}$. Le signal $\overline{\text{PSEN}}$ est activé deux fois par cycle machine et correspond aux deux récupérations de codes. Dans la figure 4.7(b), où un accès à la mémoire externe de données est exécuté (MOVX), on remarque la suppression des deux activations de $\overline{\text{PSEN}}$ correspondant à la suppression des deux récupérations de codes durant le second cycle machine.

Lorsque le programme est récupéré en ROM ou EPROM interne, le signal $\overline{\text{PSEN}}$ reste inactif. Par contre le signal ALE reste opérationnel dans tous les cas de figure. Il peut constituer un signal dont la fréquence est stable à condition qu'aucun accès à une mémoire externe de données ne soit programmé : la figure 4.7(b) permet de remarquer la suppression d'un signal ALE pour les instructions du type MOVX. En l'absence de ce type d'instruction la sortie ALE peut constituer un signal d'horloge de fréquence six fois plus faible que celle de l'oscillateur du microcontrôleur.

Initialisation du microcontrôleur

L'entrée RST permet d'initialiser le microcontrôleur. Cette entrée est dotée d'un trigger de Schmitt.

Une réinitialisation est correctement effectuée si l'entrée RST est maintenue à l'état 1 pendant au moins deux cycles machine, soit 24 périodes de l'oscillateur, après que celui-ci ait été mis en service. L'entrée RST est échantillonnée sur la phase S5P2 de chaque cycle machine. Les broches des ports conservent leur état logique pendant une durée correspondant à 19 périodes de la fréquence de l'oscillateur après la phase S5P2 où un état logique 1 a été détecté sur RST. Une ligne de port orientée en sortie peut donc conserver son état durant un délai variant de 19 à 31 périodes d'oscillateur après l'apparition d'un ordre d'initialisation.

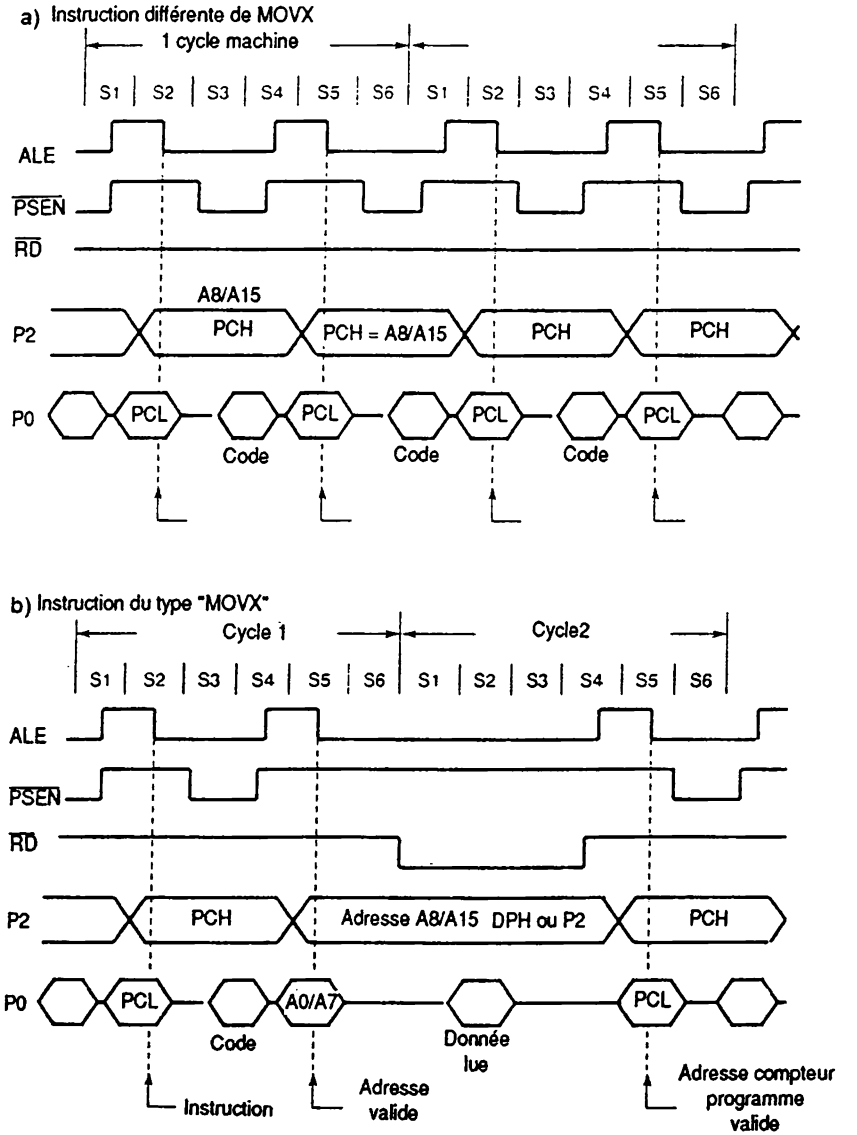


Figure 4.7 : Chronologie des signaux lors de la récupération d'un code ou d'une donnée depuis une mémoire externe

Tant que l'entrée RST est à 1, les signaux de contrôle ALE et $\overline{PSEN}$ sont forcés à l'état 1. Lorsque RST repasse à l'état inactif 0, il faut compter un ou deux cycles machine avant que ces signaux soient correctement cadencés.

La séquence d'initialisation remet à 0 tous les registres à fonction spéciale exceptés :

les quatre registres P0, P1, P2 et P3 correspondant aux ports qui eux sont forcés à 1 ;

le registre pointeur de pile initialisé à la valeur 07 ;

le registre SBUF (tampon de communication série) dans un état indéterminé ;

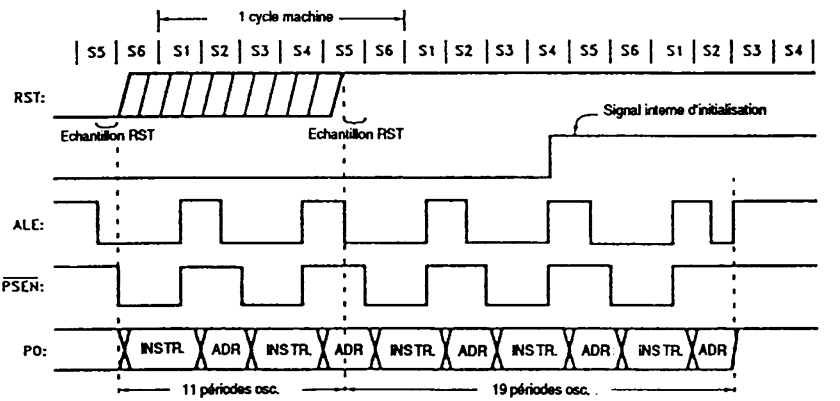

1 cycle machine
S5 S6 S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6 S1 S2 S3 S4
RST:
Echantillon RST
Echantillon RST
Signal interne d'initialisation
ALE:
PSEN:
P0:
INSTR. ADR INSTR. ADR INSTR. ADR INSTR. ADR INSTR. ADR
11 périodes osc.
19 périodes osc.

Figure 4.8 : Chronogramme d'initialisation

La notion de **registre à fonction spéciale** (SFR) est définie et détaillée dans le chapitre consacré à l'étude de la mémoire vive interne. Les registres de la RAM interne n'ayant pas de fonction spéciale ne sont pas modifiés par une opération d'initialisation. Une initialisation correcte est assurée à la mise sous tension en réalisant le câblage décrit sur la figure 4.9. Pour les versions CMOS, la résistance de 8,2 kΩ peut être omise et la capacité peut être réduite à 1 μF.

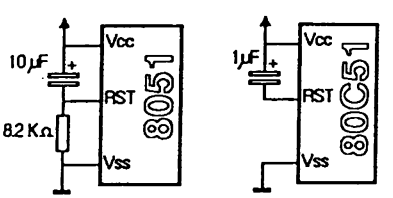


Figure 4.9 : Circuits d'initialisation standards

5

Mémoire programme et mémoire de données

Séparation logique entre programme et données

Tous les microcontrôleurs de la famille 8051 sont caractérisés par la particularité de pouvoir adresser d'une façon séparée un espace **mémoire de programme** et un espace **mémoire de données**. Les différents plans mémoire sont représentés par la figure 5.1. pour les versions 8051 et par la figure 5.2. pour les 8052.

Par mémoire de programme on désigne la mémoire dans laquelle se trouve le code machine exécuté par le processeur, par opposition à la mémoire de données où se trouvent les données manipulées par le processeur. Cette séparation de fonction se traduit par des modes d'accès et des adressages différents. La mémoire programme n'est accessible qu'en lecture et dans le cas où le programme est logé dans un composant extérieur, un seul signal particulier ($\overline{PSEN}$) permet de valider la lecture de ce composant. La mémoire programme est toujours adressée sur 16 bits.

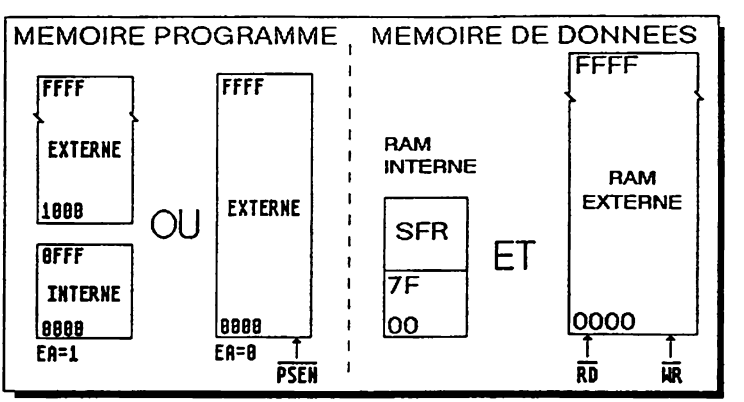


Figure 5.1 : Organisation des espaces mémoires des 8051

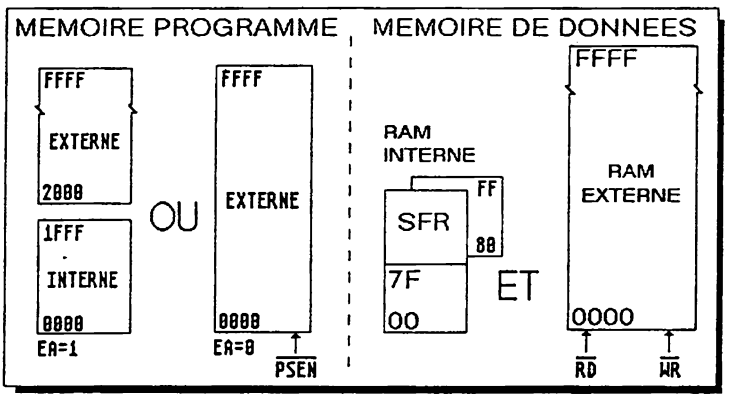


Figure 5.2 : Organisation des espaces mémoires des 8052

La mémoire de données est accessible aussi bien en lecture qu'en écriture. Une RAM interne est prévue à cet effet. Si la capacité de celle-ci s'avérait insuffisante, il est possible de compléter l'espace de mémoire de données grâce à une RAM externe. Le microcontrôleur peut commander cette mémoire en lecture et écriture à l'aide des signaux $\overline{RD}$ et $\overline{WR}$. L'adressage de cette mémoire peut s'envisager par bloc de 256 octets (adressage sur 8 bits) ou directement par adressage sur 16 bits.

Le microcontrôleur est en premier lieu conçu pour se suffire à lui-même. L'accès à des mémoires externes est donc considéré comme une option. Malgré cette restriction les caractéristiques de vitesse sont conservées en ce qui concerne l'accès à un programme. Par contre l'accès à une donnée externe nécessitera dans tous les cas un temps d'exécution de deux cycles machine, soit 2 μs lorsque la fréquence d'horloge est de 12 MHz.

Mémoire programme

La mémoire programme n'est accessible qu'en lecture, et est logée dans une mémoire morte. Dans les versions intégrant une ROM ou EPROM, celle-ci est placée en début de l'espace d'adressage de 64 Koctets. Dans les versions non équipées de ROM interne, l'accès au programme externe est assuré par le signal de commande $\overline{PSEN}$ (Programm Store ENable).

L'espace inférieur (4K, 8K, ou 16Koctets) de la mémoire programme peut être occupé par la ROM interne ou remplacé par une EPROM extérieure. La sélection est réalisée en fonction de l'état logique présent sur la broche $\overline{EA}$ (External Access).

Durant la phase d'initialisation du microcontrôleur, la broche $\overline{EA}$ (External Address) est testée. En fonction de l'état de cette entrée, les codes instructions sont recherchés soit en mémoire interne pour $\overline{EA}$ = 1, soit en mémoire externe si $\overline{EA}$ = 0.

Microcontrôleur équipé d'une ROM interne

Selon la version choisie, la partie inférieure de l'espace mémoire programme est occupée par une ROM ou EPROM de 4 Koctets, 8 Koctets. Si la broche $\overline{EA}$ est forcée au niveau logique 0, cette mémoire est inactive et le microcontrôleur entreprend la recherche des instructions en mémoire externe. Cette situation présente l'intérêt de pouvoir récupérer et d'utiliser une version du microcontrôleur équipée d'une ROM.

Dans le cas plus classique où la broche $\overline{EA}$ est dans l'état 1, le programme est alors récupéré à partir de l'adresse 0 de la mémoire interne. Si le programme utilise une instruction de saut ou un appel à un sous-programme en spécifiant une adresse de destination supérieure à la capacité de la mémoire interne, la recherche des codes instruction se poursuit alors automatiquement en mémoire programme externe. Cette remarque est valable pour la plupart des versions des microcontrôleurs :

Pour exemple, un 8751 doté de 4 Koctets d'EPROM, le programme est récupéré dans l'EPROM interne de l'adresse 0 à l'adresse 0FFFH puis éventuellement dans la mémoire externe de 1000H à FFFFH.

Un 8052 doté de 8 Koctets de ROM, le programme est récupéré dans la ROM interne de l'adresse 0 à l'adresse 1FFFH puis en mémoire externe de 2000H à FFFFH.

Il faut quand même noter au moins une exception : le 8051 AHP limité à un adressage de 4 Koctets. Cette limitation est volontaire et constitue une protection du programme placé en ROM interne.

Le signal de lecture de la mémoire programme externe ($\overline{PSEN}$) reste inactif lors d'une récupération de code depuis la mémoire programme interne.

Microcontrôleur sans mémoire programme interne

Pour les microcontrôleurs dépourvus de mémoire programme interne, la broche $\overline{EA}$ doit être obligatoirement forcée au potentiel V_{SS} pour que les instructions soient recherchées en mémoire externe dès l'adresse 0.

La figure 5.3 présente une configuration matérielle de principe pour un microcontrôleur exécutant un programme logé dans une EPROM externe. Le signal $\overline{PSEN}$, qui n'est actif que pour les accès à la mémoire programme, sert à valider le tampon de sortie de l'EPROM ($\overline{OE}$).

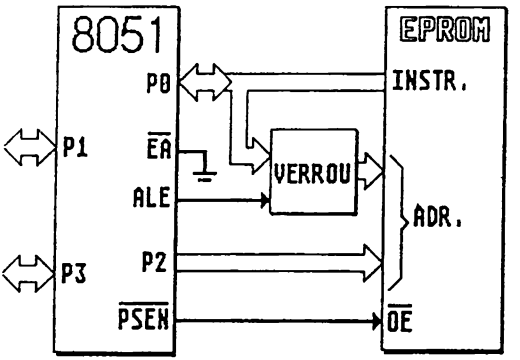


Figure 5.3 : Connexions entre microcontrôleur et mémoire de programme externe

Un circuit "verrou" permet de démultiplexer adresse et donnée : le port P0 présente la partie basse de l'adresse du compteur programme (PCL). Cette adresse est mémorisée sur les sorties du verrou grâce au signal ALE. Dans le même temps le port P2 présente la partie haute de l'adresse contenue dans le compteur programme (PCH). Puis le port P0 se place dans un état de haute impédance et peut ainsi recevoir le code instruction lorsque $\overline{PSEN}$ passe à 0 validant la sortie de l'EPROM. Lorsque le port P0 est utilisé comme bus d'adresses et de données le registre à fonction spéciale lui correspondant est automatiquement modifié par l'unité centrale. Tous les bits de ce registre sont mis à 1. Cette précision n'est valable que pour le registre P0. En effet, le port P2 n'est modifié que momentanément lors du cycle de récupération d'une instruction. Il ne présente la valeur haute du compteur programme (PCH) que pendant la période de récupération du code, puis la valeur du registre P2 est remise en place automatiquement.

Utilisation de l'espace mémoire programme

Après une opération d'initialisation (RESET), le compteur programme contient la valeur 0. C'est donc à cette adresse que le processeur récupère la première instruction à exécuter. Comme le montre le tableau 5.4, dans la partie inférieure de l'espace mémoire programme, une zone est dédiée aux services des interruptions. Les adresses de cette zone correspondent aux vecteurs des différentes sources d'interruption : la source d'interruption, si elle est autorisée par l'utilisateur, provoque un saut à une de ces adresses. Par exemple, si après autorisation, l'interruption extérieure de niveau 0 survient, le processeur termine l'exécution de l'instruction en cours, puis le compteur programme se voit affecté de la valeur 0003. Le processeur exécutera le programme d'interruption qui devra donc être logé à partir de cette adresse. Si le processus d'interruption n'est pas utilisé dans l'application envisagée, le début de l'espace mémoire programme peut être attribué de nouveau au programme

principal. La zone allouée à chaque service d'interruption est de 8 octets. Si cet espace est insuffisant pour recevoir le sous-programme complet, celui-ci peut se poursuivre, après instruction de saut, à une adresse quelconque de l'espace mémoire programme.

adresse	source interruption	symbole
0000	RESET	
0003	interruption externe	IE0
000BH	Timer 0	TF0
0013H	interruption externe	IE1
001BH	Timer 1	TF1
0023H	port série	RI/TI
002BH	Timer 2 (8052 seulement)	TF2/EXF2

Figure 5.4 : Zone de la mémoire programme dédiée aux interruptions

Un programme utilisant les sources d'interruption doit donc débuter par une instruction de saut placée à l'adresse 0 et désignant une adresse destination au-delà de la zone dédiée aux sous-programmes d'interruption. L'espace de trois octets entre les adresses 0 et 2 est suffisant pour recevoir le code et l'adresse de ce saut.

Mémoire de données

La partie droite des figures 5.1 et 5.2 fait apparaître l'espace de mémoire vive interne au microcontrôleur et l'espace de mémoire externe que ce microcontrôleur peut adresser.

Mémoire vive interne

Cette mémoire se décompose en trois parties de 128 octets chacune. Le bloc de 128 octets situé de l'adresse 0 à l'adresse 7FH constitue la partie basse de cette mémoire interne et il est présent dans tous les microcontrôleurs de la famille MCS 51. Il en est de même pour le bloc nommé SFR. L'appellation SFR se traduit par Registres à Fonction Spéciale. Le nombre de ces registres peut varier selon le modèle de microcontrôleur. Parmi ces registres on trouve, entre autre, les registres P0, P1, P2 et P3 correspondant aux 4 ports de même nom, l'accumulateur A, etc. La liste et l'étude de ces registres sont développées dans le chapitre 8.

Le troisième bloc de mémoire situé entre les adresses 80H et FFH n'existe que sur les versions 8052. Sur la figure 5.5 ce bloc de mémoire et les registres SFR sont placés en superposition car ces deux zones occupent le même plan d'adresses. La distinction entre ces deux blocs reste possible par restriction des modes d'adressages possibles. Les registres à fonction spéciale ne sont accessibles qu'en adressage direct alors que la zone mémoire n'est accessible que par adressage indirect par registre. Cette disposition particulière permet d'adresser globalement 384 octets, au lieu de 256, grâce à un bus d'adresses de 8 bits.

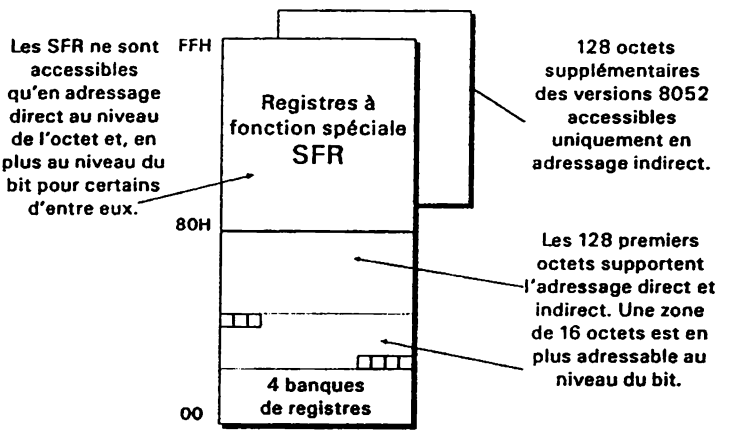

Les SFR ne sont accessibles qu'en adressage direct au niveau de l'octet et, en plus au niveau du bit pour certains d'entre eux.
FFH
Registres à fonction spéciale SFR
80H
4 banques de registres
00
128 octets supplémentaires des versions 8052 accessibles uniquement en adressage indirect.
Les 128 premiers octets supportent l'adressage direct et indirect. Une zone de 16 octets est en plus adressable au niveau du bit.

Figure 5.5 : Organisation générale de la mémoire interne de données

Accès à des données externes

La figure 5.6 donne une organisation de principe des connexions entre microcontrôleur et une mémoire RAM utilisée comme mémoire externe de données. La structure est simplifiée et l'on considère dans ce cas que le microcontrôleur exécute un programme situé en ROM interne.

Le principe de démultiplexage du port P0 est identique à celui de la figure 5.3. Par contre le contrôle de la mémoire de données externe est assuré par les lignes P3.7 et P3.6, baptisées pour l'occasion $\overline{RD}$ et $\overline{WR}$. Le signal $\overline{RD}$, actif à l'état bas, assure une opération de lecture de la RAM externe alors que le signal $\overline{WR}$, actif à l'état bas, assure une écriture dans cette RAM. Ces signaux sont inactifs lors d'un accès à une mémoire externe de programme. Pour la récupération d'un code instruction l'adressage était implicitement d'un format de 16 bits et mobilisait les ports P0 et P2. Pour l'adressage de la mémoire externe de données il est possible d'envisager deux formats d'adressage. Au niveau logiciel, ces adressages sont du type indirect par pointeur.

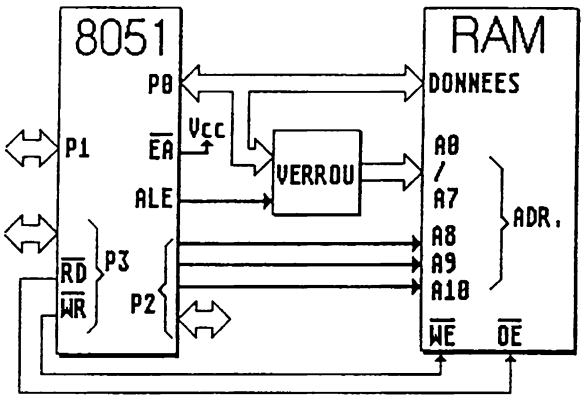


Figure 5.6 : Microcontrôleur et mémoire externe de données

L'adressage de la RAM externe peut être limité à un espace de 8 bits. On utilise un des deux registres R0 ou R1 comme pointeurs 8 bits. Le programmeur doit placer l'adresse dans un de ces deux registres et le transfert de données s'effectue par l'intermédiaire de l'accumulateur. Le port P0 présente l'adresse (A0, A7) puis envoie ou reçoit la donnée. Si la RAM externe est d'une capacité supérieure à 256 octets, il est possible d'envisager un adressage par "*bloc*" ou par "page" de 256 octets. les broches d'adresse A8, A9, etc. du circuit de RAM sont reliées aux lignes P2.0, P2.1, etc. L'adressage du bloc est alors défini en plaçant dans le registre P2 le numéro de la *page*, et l'adressage dans la page est défini par le contenu du registre R0 (ou R1).

L'adressage sur 16 bits reste possible, la mémoire est alors adressée grâce au pointeur de 16 bits nommé DPTR. La partie basse de ce pointeur (DPL) est placée sur P0, et la partie haute (DPH) sur P2. Il est concevable de mélanger les deux types d'adressage : une première instruction de transfert désigne la mémoire par adressage sur 16 bits, puis les transferts suivants sont définis par adresses de 8 bits. Le premier adressage sur 16 bits réalisé à partir du pointeur DPTR modifie le registre du port P2, on fixe alors le bloc de 256 octets sur lequel les transferts vont avoir lieu. Bien que le jeu d'instructions ne soit présenté que dans les chapitres suivants, un exemple de programme illustrant cette possibilité est présenté ci-dessous :

```
MOV DPTR,#4000H
le pointeur 16 bits est affecté de la valeur 4000H
MOV R0,#01
le registre R0 pointeur 8 bits prend la valeur 01
MOVX A,@DPTR
```

la donnée située dans la RAM à l'adresse 4000H est placée dans l'accumulateur. On remarquera la syntaxe spéciale MOVX permettant de spécifier au processeur un adressage de mémoire externe.

MOV R4,A
la donnée est stockée dans un registre pour libérer A
MOVX A,@R0
la donnée située en RAM externe à l'adresse 4001H est placée dans A. R0 désigne la partie basse 01 et P2 a conservé la partie haute 40H.

Si une application nécessite une mémoire externe de données supérieure à 64 Koctets, il est possible d'étendre le bus d'adresses par l'utilisation de lignes du port P1 ou du port P3. On réalise ainsi un plan mémoire composé de plusieurs pages de 64 Koctets. Le pointeur DPTR permet alors l'adressage dans la page active. Le changement de page est à gérer par l'utilisateur.

Exemple d'adressage de mémoires externes

Le schéma structurel de la figure 5.7 propose une organisation matérielle autour d'un microcontrôleur du type 8032. Cette version n'étant pas équipée de ROM interne, nécessite donc l'adjonction d'une EPROM (2764). Pour compléter l'exemple une mémoire externe de données de 8 Koctets (RAM 6264) a été prévue. Un circuit 74LS573 (ou HCT573) est utilisé pour le démultiplexage adresses/données. Cette situation extrême, où le microcontrôleur ne forme plus un ensemble se suffisant à lui même, permet de constater malgré tout certains avantages par rapport à l'utilisation d'un microprocesseur classique : l'ensemble dispose de 12 lignes utilisables comme entrées ou sorties, une interface de communication série avec l'environnement, et sans oublier les timers intégrés dont les entrées T0, T1 et T2 pourraient être utilisées pour des opérations de comptage. L'adressage de la mémoire 6264 reste ambigu car les bits A14 et A15 ne sont pas pris en compte. Cette situation n'est pas gênante si la structure est considérée comme définitive.

Mémoire à fonction mixte

Dans une application où une partie du programme doit être souvent modifiée, il est envisageable de placer, dans une EPROM, la partie du programme non modifiable, et de placer son complément, sujet à modification, dans une mémoire RAM. Pour charger le programme dans cette mémoire, il faut qu'elle soit accessible en tant que mémoire de données, alors que dans la phase d'exécution de ce programme, il faut qu'elle soit reconnue par le microcontrôleur comme une mémoire programme. La figure 5.8 donne un exemple de solution à ce problème. Au niveau logique la fonction mixte est assurée par un ET logique entre les signaux $\overline{PSEN}$ et $\overline{RD}$.

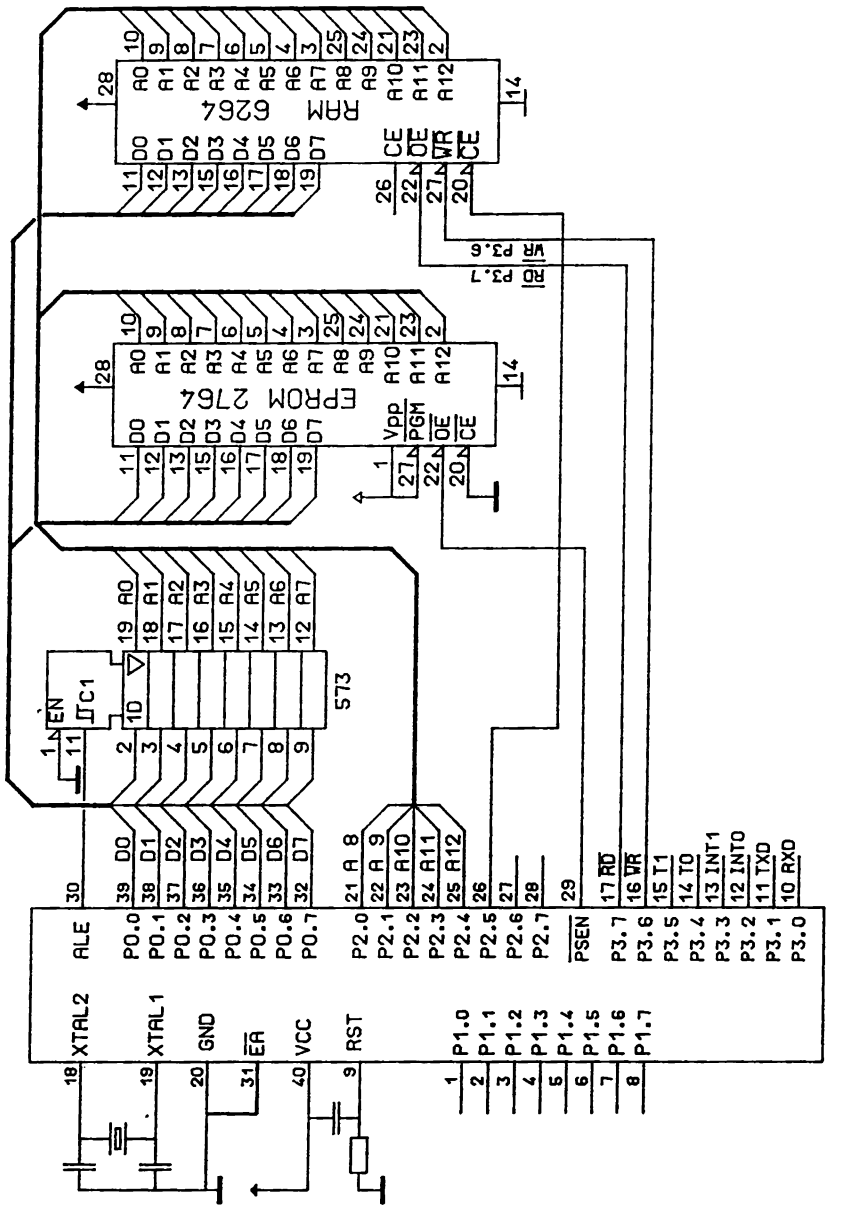


Figure 5.7 : Exemple d'organisation matérielle autour d'un 8032

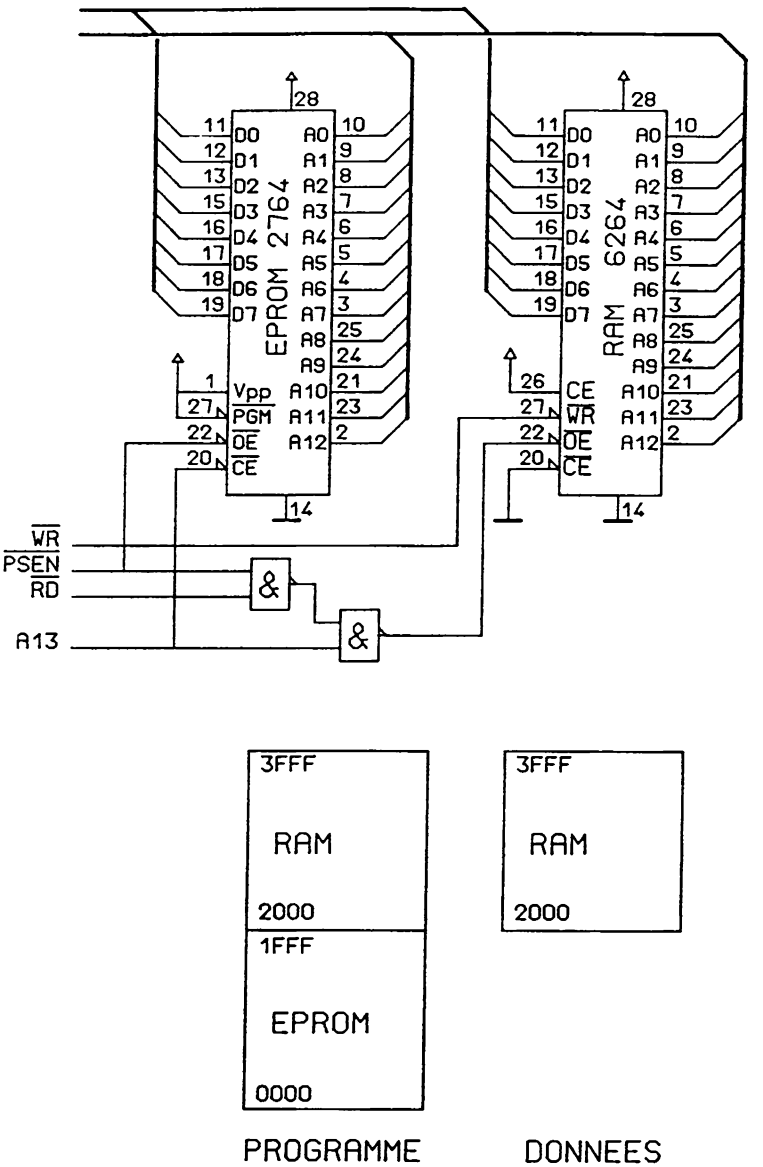


Figure 5.8 : Une RAM externe en mémoire de programme et de données

6

Modes d'adressage et jeu d'instructions

Introduction

Tous les membres de la famille MCS-51 exécutent le même jeu d'instructions. Ces instructions ont été optimisées pour convenir à la commande et au contrôle de processus se suffisant d'une architecture 8 bits. Il autorise de nombreux modes d'adressage permettant entre autres des accès rapides aux données logées en RAM interne facilitant ainsi des opérations portant sur de petites structures de données. Cette organisation du type processeur de données 8 bits est complétée par la possibilité de manipuler directement des variables réduites au format du bit (variables booléennes). Le constructeur parle d'un *processeur booléen* pour définir cette possibilité supplémentaire. Ce processeur n'est en fait, vu de l'utilisateur, qu'un mode d'adressage supplémentaire. Le processeur booléen utilise l'indicateur de report repéré par le symbole Cy comme accumulateur. On peut alors effectuer toutes les opérations logiques classiques directement entre cet accumulateur et des variables binaires en évitant les opérations préalables d'isolement. Cette possibilité est mise en évidence dans le chapitre *processeur booléen.*

En introduction à la description du jeu d'instructions, il est proposé dans les paragraphes qui suivent, d'une part, de préciser les modes d'adressage possibles ainsi que leur syntaxe et, d'autre part, de définir les groupes d'instructions existant pour cette famille de microcontrôleurs.

Modes d'adressages

Le jeu d'instructions utilise six modes d'adressages qui sont les suivants :

Adressage immédiat

Au niveau code machine ce mode d'adressage se traduit par le fait que le code opération est suivi d'un deuxième octet représentant une donnée. Du fait que cette

donnée est logée dans la mémoire programme, en général mémoire seulement accessible en lecture, elle constitue donc par définition une constante. Au niveau de la syntaxe dans le code source, l'adressage immédiat utilise le caractère # .

Exemple :

Placer dans A la valeur 30H :

```
MOV A,#30H
```

placer dans A la valeur fixée au symbole VAL :

```
MOV A,#VAL
```

Adressage direct

Un des opérandes de l'instruction est une adresse sur le contenu de laquelle va porter l'opération. Aucun signe particulier n'est employé dans la syntaxe du code source.

Exemple : MOV A,30H
signifie placer dans A la donnée située à l'adresse 30H.

Ce mode d'adressage est le seul moyen d'accès aux registres à fonction spéciale. D'une façon générale, l'adressage direct ne concerne que les 128 octets de poids faible de la RAM interne et les SFR.

Adressage par registre

Le (ou les) opérande(s) de l'opération sont représentés par des noms de registres. En fait l'opération porte sur les valeurs contenues dans ces registres.

Exemple :

Transfert dans A de la valeur contenue dans R1 :

```
MOV A, R1
```

additionner à cette valeur le contenu de R0

```
ADD A, R0
```

Adressage indirect par registre

L'adressage indirect par registre utilise le contenu des registres R0 ou R1 comme une adresse 8 bits. Ces pointeurs permettent de travailler sur des données situées dans les

256 premiers octets de la RAM interne (128 octets pour les 8051/31/8751), ou sur une page de 256 octets de la mémoire externe. Dans ce cas la page est définie par le contenu du PORT P2. Ce mode d'adressage ne permet pas un accès aux registres à fonction spéciale (SFR). Le signe @ est utilisé dans le code source pour indiquer à l'assembleur ce mode d'adressage.

Exemple :

Placer dans R0 la valeur 30H par adressage immédiat :

```
MOV R0,#30H
```

le contenu de l'adresse 30H est placé dans A

```
MOV A, @R0
```

Le même type d'adressage est possible avec le registre 16 bits DPTR ; l'adressage porte alors sur les 64 Koctets de la mémoire externe. Les instructions de sauvegarde sur la pile et de récupération depuis la pile utilisent implicitement ce mode d'adressage grâce au pointeur SP.

Adressage indirect indexé par registre

Ce mode d'adressage ne permet d'accéder qu'en mode lecture à la zone de mémoire programme. Il permet une récupération de données structurées sous forme de table placée dans le code programme.

Dans ce mode d'adressage, le compteur programme PC ou le pointeur de données DPTR servent de pointeur. Le déplacement ajouté à la valeur de ce pointeur est le contenu de l'accumulateur, et la valeur récupérée est placée dans l'accumulateur. Ce mode d'adressage facilite l'accès à des tables de données mais la valeur du déplacement est perdue.

Exemple : On ajoute 1 à la valeur de A pour compenser l'instruction RET :

```
INC A
```

puis A prend la valeur 50H si A valait 1 avant appel de ce sous-programme :

```
MOVC A,@ A +PC
RET
DB 50H
DB 60H
etc.
```

Adressage au niveau du bit

Comme il en est fait mention en début de ce chapitre, les microcontrôleurs de la famille MCS-51 intègrent un processeur booléen pour lequel le rôle d'accumulateur est assuré par l'indicateur de report CY nommé pour l'occasion C. Un ensemble d'instructions permet des opérations logiques portant sur l'état de bits. Un adressage au niveau du bit est donc possible. L'adresse étant spécifiée sur un octet, le nombre de bits directement adressables est donc limité à 256. L'adresse de ces bits est définie d'une part, dans le tableau des registres à fonction spéciale, d'autre part dans le tableau détaillant la RAM interne.

Exemple :

Placer dans CY l'état du bit 3 du PORT 1 :

```
MOV C, P1.3
```

ET logique avec le bit 7 de la RAM d'adresse 22H :

```
ANL C, 17H
```

OU logique avec complément du bit 0 du PORT P2 :

```
ORL C,/P2.0
```

et ranger le résultat dans le bit 7 de la RAM d'adresse 2FH

```
MOV 7FH,C
```

Les opérations au niveau du bit sont la mise à 1, la mise à 0, le complément, le saut conditionnel à l'état du bit, le saut conditionnel à l'état 1 d'un bit et la remise automatique à 0 de ce bit, le ET logique, le OU logique etc.

Classification des instructions

Le nombre d'instructions exécutables par un des microcontrôleurs de la famille MCS-51 est de 256. Ce nombre prend en compte toutes les formes possibles d'une même instruction. Ces instructions peuvent se classer en 5 groupes :

Instructions de transfert

La forme générale utilisée précise en premier lieu la destination du transfert puis la source de la valeur à transférer. Le tableau 6.1 résume la forme générale des instructions utilisables sur la RAM interne. Toutes ces instructions sont exécutées en un ou deux cycles machine. Pour une fréquence d'oscillateur de 12 MHz le temps d'exécution est donc au maximum de 2 μs

Les instructions de la forme MOV <dest>,<src> réalisent un transfert entre deux mémoires de la RAM interne sans utiliser l'accumulateur comme registre de transit.

Il faut garder présent à l'esprit que la pile doit obligatoirement résider en RAM interne ce qui justifie la présence des instructions PUSH et POP dans le tableau 6.1. Ces instructions n'utilisent que le mode d'adressage direct. Il est donc impératif de désigner par son adresse la valeur à sauvegarder dans la pile (ou indiquer l'adresse de destination lorsqu'on récupère une valeur depuis la pile). Cette remarque prend toute son importance lorsqu'on désire sauvegarder le contenu d'un registre de la banque active (R0 à R7).

Mnémonique	Dir	Ind	Reg	Imm	cycle
MOV A,<src>	x	x	x	x	1
MOV <dest>,A	x	x	x		1
MOV <dest>,<src>	x	x	x	x	2
MOV DPTR,#data16				x	2
PUSH <src>	x				2
POP <dest>	x				2
XCH A,<byte>	x	x	x		1
XCHD A,@Ri		x			1

Tableau 6.1 : Instructions de transfert utilisables sur l'espace de la RAM interne

Exemple : L'instruction PUSH R1 provoquera une erreur lors de l'assemblage d'un programme car R1 ne peut pas être un symbole représentant une adresse. La syntaxe légale sera PUSH 01.

Il est possible de charger une valeur de 16 bits dans le pointeur DPTR par l'instruction MOV DPTR,#data16. Ce pointeur pourra être utilisé pour manipuler des données situées aussi bien dans l'espace de mémoire programme que dans l'espace de mémoire externe.

L'instruction XCH A,<byte> permet de permuter les valeurs de l'accumulateur et de la mémoire désignée par son adresse 8 bits. L'instruction XCHD A,@Ri provoque aussi une permutation mais limitée à la moitié de l'octet. Seul le quartet de poids faible de l'octet est échangé par cette instruction. Le tableau 6.2. regroupe les instructions de

transfert spécifiques à la mémoire RAM externe. Le transfert entre cette mémoire et le microcontrôleur utilise dans tous les cas l'accumulateur comme source ou comme destination. La lettre X est ajoutée en suffixe à la mnémonique de base MOV :

MOVX = MOV EXTERNAL

Format	MNEMONIQUE	cycle
8 bits	MOVX A, @Ri	2
8 bits	MOVX @Ri, A	2
16 bits	MOVX A, @DPTR	2
16 bits	MOVX @DPTR, A	2

Tableau 6.2 : Instructions de transfert avec la RAM externe

L'accès aux données situées en mémoire externe ne peut s'effectuer qu'en adressage indirect. Par contre il est possible de choisir deux formats de pointeur :

— adressage sur 8 bits grâce aux pointeurs R0 ou R1,

— adressage sur 16 bits à l'aide du pointeur DPTR.

Les instructions de transfert avec la RAM externe activent implicitement soit un signal de lecture (fonction auxiliaire de la broche P3.7), soit un signal d'écriture (fonction auxiliaire de P3.6).

Une autre forme de transfert est réalisable. Il s'agit de récupérer une donnée constante appartenant à une table logée en mémoire programme. La mnémonique utilisée est alors MOVC pour move constant. Deux instructions sont dédiées à ce genre d'opération ; Il s'agit de :

MOVC A, @A+DPTR

et MOVC A, @A+PC

L'accumulateur est chargé d'une valeur représentant la position de la donnée dans la table. Cette valeur est ajoutée au pointeur (DPTR ou PC), puis la valeur de la donnée ainsi désignée est placée dans l'accumulateur. La table peut contenir jusqu'à 256 éléments. L'instruction qui utilise PC comme pointeur oblige à placer la structure de table à la suite la plus immédiate possible de l'instruction.

Exemple : Dans l'exemple qui suit on place le numéro de la donnée à récupérer puis on appelle le sous-programme qui va se charger de récupérer la donnée. Au retour d'exécution du sous-programme l'accumulateur est chargé de la donnée.

```
        MOV     A, #NUMERO    ;NUMERO=3
        CALL    TABLE
                              ;A=30H

TABLE:  INC     A
        MOVC    A,@A+PC
        RET
        DB      10H
        DB      20H
        DB      30H
```

Dans le cas où A vaut 0, la valeur récupérée correspond au code de l'instruction RET. Il est difficile d'éviter de placer au moins une instruction après l'ordre de transfert : soit un ordre de retour, soit un ordre de saut afin que la table ne soit pas considérée comme des codes exécutables ! Pour compenser cette instruction de rupture de séquence il suffit de compenser la valeur de A avant calcul du déplacement. Dans l'exemple ci-dessus l'instruction INC A ajoute systématiquement 1 à la valeur de A et permet ainsi de considérer la donnée 10H comme la donnée d'indice 0. Cette façon de faire limite la taille de la table à 255 éléments : de 0 à 254. Il est aussi possible d'omettre la compensation INC A et d'interdire l'indice de table 0.

Instructions arithmétiques

Le tableau 6.3. donne un aperçu de la variété des instructions arithmétiques exécutables. Une addition avec le contenu de l'accumulateur est réalisable selon 4 modes d'adressage. Le mnémonique générale ADD A,<byte> peut prendre par exemple les formes :

```
ADD     A,7FH   adressage direct
ADD     A,@R0   adressage indirect
ADD     A,R6    adressage par registre
ADD     A,#7FH  adressage immédiat
```

La mnémonique ADDC permet de compléter cette addition par la valeur de l'indicateur de report.

Mnémonique	Dir	Ind	Reg	Imm	cycle
ADD A,<byte>	x	x	x	x	1
ADDC A,<byte>	x	x	x	x	1
SUBB A,<byte>	x	x	x	x	1
INC A					1
INC <byte>	x	x	x		1
INC DPTR					2
DEC A					1
DEC <byte>	x	x	x		1
MUL AB					4
DIV AB					4
DA A					1

Tableau 6.3 : Instructions arithmétiques

La soustraction entre la valeur de l'accumulateur et un opérande source prend obligatoirement en compte l'état de l'indicateur. Cette instruction est donc orientée vers une simplification d'une opération en multiple précision, mais elle nécessite de contrôler l'état de l'indicateur en début de soustraction. Ainsi une soustraction en simple précision doit débuter par une remise à 0 de l'indicateur de report.

Il est possible d'incrémenter ou de décrémenter directement un octet de mémoire interne sans passer par l'accumulateur, mais ces opérations ne positionnent pas les indicateurs. Une instruction permet aussi d'incrémenter le pointeur 16 bits DPTR.

L'instruction MUL AB permet une multiplication 8 bits entre l'accumulateur et le registre B. Le résultat de 16 bits est obtenu par concaténation des contenus de B (poids fort) et A (poids faible). Cette opération s'exécute en quatre cycles machine soit un temps de 4 μs pour une horloge cadencée à 12 MHz.

Pour un même temps d'exécution il est possible avec l'instruction DIV AB de réaliser une division de A par B. Le quotient sera rangé dans A et le reste dans B.

La dernière instruction (DA A) permet l'ajustement décimal. Lors d'une addition de deux valeurs codées en BCD il est nécessaire d'utiliser cette instruction sur le résultat afin que celui-ci soit aussi en code BCD. Il faut noter que cette instruction n'est pas

prévue pour transformer une valeur codée BINAIRE en valeur codée BCD mais seulement pour ajuster un résultat d'une opération effectuée sur deux valeurs déjà codée BCD.

Instructions logiques

La liste des instructions logiques est donnée dans le tableau 6.4. Les opérations logiques habituelles (ET, OU, OU exclusif, complément) portant sur un octet sont réalisées bit à bit en un temps d'exécution d'un cycle machine si l'un des opérandes est l'accumulateur.

Mnémonique	Dir	Ind	Reg	Imm	cycle
ANL A,<byte>	x	x	x	x	1
ANL <byte>,A	x				1
ANL <byte>,#data	x				2
ORL A,<byte>	x	x	x	x	1
ORL <byte>,A	x				1
ORL <byte>,#data	x				2
XRL A,<byte>	x	x	x	x	1
XRL <byte>,A	x				1
XRL <byte>,#data	x				2
CLR A					1
CPL A					1
RL A					1
RLC A					1
RRL A					1
RLC A					1
RR A					1
RRC A					1
SWAP A					1

Tableau 6.4 : Instructions logiques

L'instruction XRL <byte>,#data constitue par exemple un moyen direct d'inverser les sorties d'un port.

Exemple :

```
XRL P1,0FFH
```

Dans cet exemple l'accumulateur n'est pas utilisé.

L'instruction RL permet une rotation d'un bit vers la gauche. Le bit de poids fort étant introduit à la place du bit de poids faible. L'instruction RR réalise de la même façon une rotation vers la droite. Le bit de poids faible étant placé à la place du bit de poids fort. Les instructions RLC et RRC réalisent des rotations sur 9 bits. Le bit sortant de l'accumulateur est placé dans l'indicateur de retenue Cy alors que l'état de cet indicateur est introduit en poids faible (RLC) ou en poids fort (RRC) de l'accumulateur.

L'instruction SWAP A assure en un cycle machine la permutation du quartet de poids faible avec le quartet de poids fort. Cette instruction remplace avantageusement 4 opérations de rotation. Elle peut faciliter la conversion d'un code binaire en code BCD. Dans l'exemple suivant, si la valeur contenue par l'accumulateur est reconnue comme inférieure à 100 (grâce à un test préalable) la conversion peut être réalisée de la façon suivante :

```
MOV   B,#10   ;FACTEUR DE DIVISION
DIV   AB      ;A=DIZAINE
SWAP  A       ;DIZAINE EN POIDS FORT
ADD   A,B     ;ADDITION DES UNITES
```

Instructions booléennes

Le terme booléen est utilisé pour créer une distinction avec les instructions logiques (paragraphe précédent). Les instructions booléennes assurent des opérations logiques directement au niveau du bit. Elles permettent une utilisation du microcontrôleur comme si celui-ci était un processeur mono bit. Les instructions booléennes regroupées dans le tableau 6.5. utilisent comme accumulateur l'indicateur de report Cy rebaptisé pour l'occasion C.

La RAM interne comprend 16 octets adressables au niveau du bit, soit un total de 128 adresses bits. La liste des bits directement adressables est complétée par les bits de certains registres dits à fonction spéciale. Il est par exemple possible d'adresser directement un des bits correspondant à l'un des quatre ports.

Les opérations possibles sont les fonctions logiques ET, OU, transfert entre l'accumulateur C et un bit ou le complément d'un bit. Il est aussi possible d'inverser

l'état d'un bit par CPL bit, de mettre à 1 ou à 0 un bit par SETB bit ou CLR bit. Enfin il est possible de réaliser des sauts conditionnels à l'état d'un bit ou l'état de l'accumulateur C.

Mnémonique	Opération	cycle
ANL C,bit	C=C and bit	2
ANL C,/bit	C=C and not bit	2
ORL C,bit	C=C or bit	2
ORL C,/bit	C=C or not bit	2
MOV C,bit	C=bit	1
MOV bit,C	bit=C	2
CLR C	C=0	1
CLR bit	bit=0	1
SETB C	C=1	1
SETB bit	bit=1	1
CPL C	C=not C	1
CPL bit	bit=not bit	1
JC rel	saut si C=1	2
JNC rel	saut si C=0	2
JB bit,rel	saut si bit=1	2
JNB bit,rel	saut si bit=0	2
JBC bit,rel	saut si bit=1 & bit=0	2

Tableau 6.5 : Instructions booléennes

On remarquera l'instruction

JBC bit,rel

capable de tester un bit, de réaliser un saut si celui-ci est à 1, et de le remettre à 0 !

L'absence d'opération OU exclusif peut être compensée par la suite d'instructions :

```
    MOV   C,bit1
    JNB   bit2,suite
    CPL   C
suite :
```

L'opération ainsi réalisée est un OU exclusif entre bit1 et bit2, le résultat étant stocké dans C. L'état de bit1 est placé dans C, puis un test de bit2 est effectué : si bit2 = 0 alors le résultat est égal à l'état de bit1. Sinon l'état de C doit être inversé.

Instructions pour rupture de séquence

Le tableau 6.6. regroupe les instructions de saut inconditionnel. L'instruction JMP addr comporte en réalité trois formes qui sont AJMP, LJMP et SJMP et qui ont pour différence le format de l'adresse désignant la destination du saut. Lors de l'écriture d'un programme en code source il est possible de n'utiliser que la forme générique JMP. C'est le programme assembleur qui définira alors la forme d'adressage appropriée au saut à réaliser.

Mnémonique	Opération	cycle
JMP addr	saut à addr	2
JMP @A+DPTR	saut à DPTR+A	2
CALL addr	appel à sous-programme	2
RET	retour de sous-programme	2
RETI	retour d'interruption	2
NOP	pas d'opération	1

Tableau 6.6 : Sauts inconditionnels

La forme SJMP addr utilise le mode d'adressage relatif. Elle autorise des sauts de −128 à +127 octets. L'adresse est alors représentée par un seul octet interprété comme une valeur signée qui est ajoutée en complément à deux à la valeur du compteur programme.

L'instruction JMP @A+DPTR permet de réaliser un saut sur une adresse calculée. L'adresse de destination est obtenue par la somme de la valeur 16 bits contenue par DPTR et de la valeur représentée par l'accumulateur. Il est possible de réaliser un saut différent pour chaque valeur de A.

Dans l'exemple qui suit A peut prendre les quatre valeurs 0;1;2;3 pour le traitement de quatre cas. Le pointeur DPTR est initialisé sur le début d'une table de saut. Chaque saut est codé sur deux octets grâce à l'utilisation de l'instruction AJMP. La valeur du cas est chargée dans l'accumulateur puis elle est multipliée par deux pour tenir compte du pas de la table de saut :

```
        MOV     DPTR,#TABLE
        MOV     A,CAS
        RL      A
        JMP     @A+DPTR
TABLE : AJMP    CAS0
        AJMP    CAS1
        AJMP    CAS2
        AJMP    CAS3
```

Si la mémoire CAS vaut 2, cette valeur est placée dans A puis l'instruction RL A donne A = 4. Cette valeur 4 est additionnée à DPTR et le compteur programme réalise un branchement à l'adresse où se trouve l'instruction AJMP CAS2. Cette instruction est alors exécutée provoquant un traitement du cas 2. La remarque faite sur les formes possibles pour l'instruction JMP addr est aussi valable pour l'instruction CALL. Cette instruction peut prendre une des formes LCALL ou ACALL. La forme ACALL permet un codage de l'adresse sur un seul octet mais n'autorise un branchement à un sous-programme que si le début de celui-ci est situé dans le bloc mémoire de deux Koctets dans lequel est placé l'instruction ACALL. Un sous-programme doit se terminer par l'instruction RET.

L'instruction RETI est une variante de l'instruction RET. Elle est réservée à un retour de sous-programme de service d'interruption. Son action est comparable à un RET mais elle assure en plus la mise à jour du système de contrôle de progression du service des interruptions.

Un dernier groupe d'instructions (tableau 6.7.) assure les sauts conditionnels. Toutes ces instructions utilisent un adressage relatif sur un octet, ce qui limite à -128 et +127 octets le saut de destination.

Mnémonique	Opération	cycle
JZ rel	Saut si A=0	2
JNZ rel	Saut si A<>0	2
DJNZ <byte>,rel	Décrémente et saut si <>0	2
CJNE A,<byte>,rel	Saut si A <> <byte>	2
CJNE <byte>,#data,rel	Saut si <byte> <>data	2

Tableau 6.7 : Sauts conditionnels

Le registre d'état PSW ne comporte pas d'indicateur de 0. Mais il est quand même possible d'effectuer un test sur l'accumulateur et de réaliser un saut en fonction de sa valeur. C'est le rôle des instructions JZ rel et JNZ rel.

L'instruction DJNZ permet de décrémenter l'opérande désigné et d'effectuer un saut si la valeur de l'opérande reste différente de 0. Ce type d'opération permet la gestion d'une structure répétitive. L'exemple suivant permet d'exécuter 10 fois la suite d'instructions située entre l'étiquette *boucle* et l'instruction DJNZ :

```
        MOV    COMPTEUR,#10
BOUCLE : .....
         .....
        DJNZ   COMPTEUR,BOUCLE
```

L'instruction CJNE permet aussi de contrôler l'exécution d'une boucle. Le saut est réalisé tant que les deux opérandes spécifiés ne sont pas égaux.

Instructions modifiant les indicateurs

Hormis une modification directe du registre PSW, seules les instructions regroupées dans le tableau 6.8 provoque une modification des indicateurs.

Instruction	Cy	Ov	AC	Instruction	Cy	Ov	AC
ADD	X	X	X	SETB C	1		
ADDC	X	X	X	CLR C	0		
SUBB	X	X	X	CPL C	X		
MUL	0	X		ANL C,BIT	X		
DIV	0	X		ANL C,/BIT	X		
DA	X			ORL C,BIT	X		
RRC	X			ORL C,/BIT	X		
RLC	X			MOV C,BIT	X		
CJNE	X						

Tableau 6.8 : Instructions modifiant les indicateurs

7

Description du jeu d'instructions

Dans les pages qui suivent, la description du jeu d'instructions des microcontrôleurs de la famille MCS_51 est présentée en conservant la syntaxe constructeur. Les instructions sont présentées par ordre alphabétique.

Après description de l'instruction dans sa forme générale, toutes les formes d'adressage autorisées sont énumérées. Cette liste s'accompagne du format de l'instruction (nombre d'octets occupés en mémoire) et du nombre de cycles machine nécessaire à l'exécution.

Syntaxe utilisée

Rn :	représente un des huit registres universels. Le repère *n* peut prendre une des valeurs comprises entre 0 et 7.
	exemple : R6
Ri :	représente un des registres R0 ou R1 pour un adressage indirect où le registre est utilisé comme pointeur.
src. byte :	opérande source dont le format est l'octet
dest.byte :	opérande destination dont le format est l'octet
src.bit :	opérande source dont le format est le bit
dest.bit :	opérande destination dont le format est le bit
adr16 :	adresse sur 16 bits
adr11 :	adresse sur 11 bits. Ce format d'adresse est complété par 5 bits de poids fort du **compteur programme**.
adr :	adresse sur 8 bits
#data :	constante ou donnée de 8 bits en adressage immédiat
rel :	octet représentant un saut relatif en valeur signée
direct :	représente une adresse directe sur 8 bits

ACALL adr11

Fonction : appel inconditionnel de sous-programme

Description : Le PC est d'abord incrémenté deux fois afin d'obtenir l'adresse de retour (adresse de l'instruction qui suit normalement l'instruction **"ACALL"**), cette adresse est sauvegardée sur la pile, puis le saut a lieu.

L'adresse du sous-programme est obtenue par la concaténation des 5 bits de poids fort du PC, des 3 bits de poids fort du code objet de l'instruction et des 8 bits du deuxième octet de l'instruction. Le sous-programme doit donc être situé dans le même bloc de 2 Koctets dans lequel se situe l'adresse représentée par le PC.

Exemple : SP= 07H l'étiquette SOUSPRG représente l'adresse 0345H.

Après exécution de :

ACALL SOUSPRG

située à l'adresse 0123H, SP vaut 09H, les mémoires RAM d'adresses 08H et 09H ont pour valeur respective 25H et 01H et le PC prend la valeur 0345H.

Format : 2 octets

Cycle : 2

ADD A, <src.byte >

Fonction : addition avec la retenue.

Description : L'opérande est additionné au contenu de l'accumulateur. Le résultat est rangé dans l'accumulateur. L'indicateur de retenue et l'indicateur de retenue auxiliaire sont respectivement mis à 1 pour un report se produisant à partir des bits 7 et 3, et mis à 0 en absence de report.

Pour une addition de deux entiers non signés, l'indicateur de retenue CY renseigne sur un éventuel débordement.

L'indicateur OV est mis à 1 lors d'une addition de deux entiers signés : si l'addition de deux nombres positifs entraîne un résultat négatif ; si la somme de deux nombres négatifs engendre un résultat positif.

Quatre modes d'adressage sont possibles pour l'opérande : adressage par registre, direct, indirect par registre et immédiat.

Exemple : L'accumulateur contient 0C3H (11000011 B) et le registre R0 contient 0AAH (10101010 B).

L'instruction ADD A,R0

place la valeur 6DH (01101101 B) dans A , AC est mis à 0, et l'indicateur de retenue ainsi que l'indicateur de débordement OV sont mis à 1.

Formes possibles :

ADD A, Rn	1 octet	Cycle : 1
ADD A, direct	2 octets	Cycle : 1
ADD A, @Ri	1 octet	Cycle : 1
ADD A, #data	2 octets	Cycle : 1

ADDC A, < src.byte>

Fonction : Addition avec la retenue

Description : L'opérande, l'indicateur de retenue et le contenu de l'accumulateur sont additionnés. Le résultat est rangé dans l'accumulateur. L'indicateur de retenue et l'indicateur de retenue auxiliaire sont respectivement mis à 1 pour un report se produisant à partir des bits 7 et 3, et mis à 0 en absence de report. Pour une addition de deux entiers non signés, l'indicateur de retenue CY renseigne sur un éventuel débordement.

L'indicateur OV est mis à 1 lors d'une addition de deux entiers signés : si l'addition de deux nombres positifs entraîne un résultat négatif ; si la somme de deux nombres négatifs engendre un résultat positif.

Quatre modes d'adressage sont possibles pour l'opérande : par registre, direct, indirect par registre et immédiat.

Exemple : L'accumulateur contient 0C3H (11000011 B) et le registre R0 contient 0AAH (10101010 B) avec l'indicateur de retenue à 1. L'instruction

```
ADDC A,R0
```

place la valeur 6EH (01101110 B) dans A, AC est mis à 0, et l'indicateur de retenue ainsi que OV sont mis à 1.

ADDC A, Rn	1 octet	Cycle : 1
ADDC A, direct	2 octets	Cycle : 1
ADDC A, @ Ri	1 octet	Cycle : 1
ADDC A, #data	2 octets	Cycle : 1

AJMP adr11

Fonction : Saut inconditionnel

Description : AJMP réalise un saut à l'adresse obtenue par une concaténation des 5 bits de poids fort du PC (après une double incrémentation de celui-ci) aux 3 bits de poids fort du code instruction et des 8 bits du second octet de l'instruction.

Exemple : L'étiquette SAUT représente l'adresse 0123H.

L'instruction

AJMP SAUT

est située à l'adresse 0345H et donnera au compteur programme la valeur 0123H.

Format : 2 octets

Cycle : 2

ANL <dest.byte>, <src.byte>

Fonction : Et logique

Description : ANL réalise l'opération logique ET entre la variable source et la variable destination. Les indicateurs ne sont pas modifiés. Quand l'opérande DESTINATION est l'accumulateur, l'opérande source peut être un registre, une adresse directe, une adresse indirecte par registre, ou une donnée immédiate. Quand l'opérande destination est une adresse directe, le source doit être soit une donnée immédiate, soit l'accumulateur.

Remarque : Lorsque cette instruction est utilisée pour modifier la valeur d'un port d'E/S, la valeur d'origine du port sera lue dans le tampon de sortie du port et non pas sur les broches du circuit.

Exemple : L'accumulateur contient 0C3H (11000011 B) et le registre R0 contient 0AAH (10101010 B).

L'instruction

ANL A, R0

placera la valeur 82H (10000010 B) dans l'accumulateur. Exemple : L'instruction

ANL P1,#01110011B

force à 0 les bits 7,3 et 2 du port P1

ANL A, Rn	1 octet	Cycle : 1
ANL A, direct	2 octets	Cycle : 1
ANL A, @Ri	1 octet	Cycle : 1
ANL A, #data	2 octets	Cycle : 1
ANL direct, A	2 octets	Cycle : 1
ANL direct, #data	3 octets	Cycle : 2

ANL C, <src. bit>

Fonction : ET logique entre bit et indicateur de retenue

Description : Si la valeur du bit source est un 0 logique, alors l'indicateur de retenue C est forcé à 0. Si la valeur du bit est 1 l'état de l'indicateur de retenue reste inchangé. Le signe "/" précédant l'opérande source est interprété par le programme assembleur comme un complément logique. L'opération alors réalisée est un ET logique entre le complément du bit source et l'indicateur de retenue mais l'état du bit source n'est pas modifié par cette opération.

Exemple : Dans l'exemple qui suit l'indicateur de retenue est mis à 1 : si et si seulement si P1.0 = 1 ACC.7 = 1 et OV = 0

```
MOV C, P1.0 ;
charger une E/S dans indicateur C

ANL C, ACC.7 ;
ET logique avec bit 7 de l'accumulateur

ANL C, /OV ;
ET logique avec complément de OV
```

Formes possibles :

ANL C, bit	2 octets	Cycle : 2
ANL C, /bit	2 octets	Cycle : 2

CJNE <dest.byte>, <>, El

Fonction : Comparaison et saut si non égalité.

Description : CJNE réalise une comparaison entre les deux premiers opérandes et provoque une rupture de séquence (saut) en cas d'inégalité. L'adresse du saut est obtenue par addition de la valeur signée, représentée par le troisième opérande, à la valeur du PC préalablement positionné sur l'adresse de l'instruction suivante. L'indicateur de retenue est mis à 1, si la valeur de l'opérande destination est inférieure à la valeur de l'opérande source. Dans les autres cas, l'indicateur de retenue est mis à 0.

Exemple : Le registre R7 vaut 56H. La première instruction de la séquence :

```
        CJNE R7, #66H, DIFF ; R7= 66H ?
         ..................; suite si R7 = 66H
DIFF : JC MINUS ;           saut si R7 < 66H
         ..................; suite si R7 > 66H
```

positionne à 1 l'indicateur de retenue et provoque un saut à DIFF. Par test de l'indicateur de retenue il est possible de déterminer si la valeur de R7 est inférieure ou supérieure à 66H.

Dans ce deuxième exemple

```
                MOV  A,#34H
WAIT :          CJNE A, P1, WAIT
```

l'instruction est exécutée à nouveau tant que P1 ne vaut pas la valeur 34H

CJNE A, direct, rel 3 octets Cycle : 2

Opération : (PC) ← (PC+3)

si (A) <> contenu de l'adresse directe alors (PC) ← (PC)+ offset REL ADR

si (A) < (directe) alors (C) ← 1 sinon (C) ← 0

CJNE A, #data, rel 3 octets Cycle : 2

CJNE Rn, #data, rel 3 octets Cycle : 2

CJNE @Ri, #data, rel 3 octets Cycle : 2

CLR A

Fonction : Remise à zéro de l'accumulateur

Description : Tous les bits de l'accumulateur sont mis à 0. Aucun indicateur n'est modifié.

Exemple : L'accumulateur A vaut 5CH (01011100 B) l'instruction

CLR A

donne à l'accumulateur la valeur 00H (00000000B).

Format : 1 octet

Cycle : 1

CLR bit

Fonction : Mise à zéro d'un bit

Description : Le bit spécifié est effacé (remis à 0). Aucun indicateur n'est modifié. Cette instruction peut porter sur l'indicateur de retenue ou sur un bit directement adressable (en RAM interne).

Exemple : Le tampon du port 1 vaut 5D H (01011101H) l'instruction

CLR P1.2

donne au tampon la nouvelle valeur 59H (01011001 B)

Formes possibles :

CLR C	1 octet	Cycle : 1
CLR bit	2 octets	Cycle : 1

CPL A

Fonction : Complémente l'accumulateur

Description : Chaque bit de l'accumulateur est complémenté (complément à 1). Chaque bit à 1 est mis à 0 et vice versa. Aucun bit n'est modifié.

Exemple : L'accumulateur A vaut 5CH (01011100 B) après l'instruction

```
CPL A
```

l'accumulateur aura pour valeur A3H (10100011 B).

Format : 1 octet

Cycle : 1

CPL bit

Fonction : Complémente un bit

Description : Le bit désigné est complémenté. Aucun indicateur n'est modifié. Cette instruction peut porter sur l'indicateur de retenue ou sur les bits directement adressables.

Exemple : Le port 1 est affecté de la valeur 5DH (01011101 B), la séquence

```
CPL P1.1
CPL P1.2
```

donne à ce port 1 la valeur 5B H (01011011 B)

Formes possibles :

CPL C	1 octet	Cycle : 1
CPL bit	2 octets	Cycle : 1

DA A

Fonction : ajustement décimal de l'accumulateur

Description : L'instruction ajuste la valeur des 8 bits de l'accumulateur résultant de l'addition de deux variables au format BCD (instructions ADD ou ADDC).

Si les 4 bits de poids faible b3 à b0 représentent une valeur supérieure à 9 (de XXXX1010 à XXXX1111) ou si l'indicateur AC est à 1, la valeur 6 est ajoutée pour ajuster le quartet de poids faible. Cette addition peut, si elle provoque un débordement vers le cinquième rang, positionner l'indicateur de retenue. Dans le cas contraire elle ne peut effacer cet indicateur. Ensuite, si l'indicateur de retenue est à 1 ou si le quartet de poids fort est supérieur à 9 (1010XXXX à 1111XXXX) ce quartet est alors incrémenté de 6. Un débordement vers le neuvième rang met à 1 l'indicateur de retenue. L'indicateur OV n'est pas affecté par ces opérations.

Exemple : L'accumulateur vaut 56H (01010110 B) représentant un ensemble de deux quartets au format BCD. Le registre R3 contient la valeur 67H (01100111 B) au format BCD et l'indicateur de retenue et à 1.

La séquence...

```
ADDC A,R3
DA   A
```

donne à l'accumulateur la valeur 0BEH (10111110 B), les indicateurs de retenue et demi retenue AC sont à 0 à l'issue de l'addition. L'ajustement décimal transforme cette valeur 0BEH en 24H et l'indicateur de retenue est mis à 1 car 56 + 67 + 1 = 124.

Une variable au format BCD peut être incrémentée ou décrémentée par addition des valeurs 01 ou 99H.

Exemple : Décrémentation de A=30H

```
ADD A,#99H
DA  A
```

donne A= 29H car 30 + 99 = 129

Format : 1 octet

Cycle : 1

DEC byte

Fonction : Décrément de l'opérande "byte"

Description : La variable indiquée est décrémentée de 1. Une valeur originale de 00H se traduira par le débordement 0FFH. Aucun indicateur n'est modifié.

Nota : quand l'instruction est utilisée pour modifier un port, la valeur utilisée comme valeur originale sera lue à partir du tampon de sortie du port.

Exemple : Le registre R0 contient 7FH (01111111 B). Les adresses mémoires 7EH et 7FH contiennent respectivement 00H et 40H, la séquence ...

```
DEC @R0
DEC R0
DEC @R0
```

donne au registre R0 la valeur 7EH et aux deux adresses mémoires 7EH et 7FH les valeurs 0FFH et 3FH.

Formes possibles :

DEC A	1 octet	Cycle : 1
DEC Rn	1 octet	Cycle : 1
DEC direct	2 octets	Cycle : 1
DEC @ Ri	1 octet	Cycle : 1

DIV AB

Fonction : Division

Description : Réalise la division de l'entier non signé, situé dans l'accumulateur, par l'entier non signé, situé dans B. L'accumulateur reçoit la partie entière du quotient et le registre B, l'entier représentant le reste de la division. L'indicateur de retenue et l'indicateur OV sont mis à 0.

EXCEPTION : si B vaut 00 les valeurs retournées par A et B sont non définies et l'indicateur OV est mis à 1.

Exemple : L'accumulateur contient 251 (11111011 B) et B vaut 18 (00010010 B) l'instruction ...

DIV AB

place la valeur 13 dans l'accumulateur et la valeur 17 dans B car 251 = (13 x 18) + 17

Format : 1 octet

Cycle : 4

DJNZ <byte>, rel

Fonction : Décrémente et saut si non égale à 0

Description : La variable byte est décrémentée de 1 et si le résultat est différent de 0, un saut relatif est exécuté. Une valeur originale de 0 provoque un débordement vers 0FFH. Aucun indicateur n'est modifié. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci.

Nota : Quand cette instruction est utilisée pour modifier une sortie de port, la valeur utilisée comme origine sera lue à partir du tampon de sortie de ce port et non pas sur les broches d'entrée.

Exemple : Les mémoires internes d'adresses 40H, 50H et 60H contiennent respectivement les valeurs 01H, 70H et 15H. La séquence ...

```
DJNZ 40H, ET1
DJNZ 50H, ET2
DJNZ 60H, ET3
```

provoque un saut à l'étiquette ET2 avec les valeurs 00H, 6FH et 15H pour les trois mémoires citées. Le premier saut vers ET1 n'a pas été exécuté car le résultat était égal à 0. Cette instruction peut être un moyen simple pour réaliser une boucle. L'exemple ci-dessous permet de réaliser une suite d'impulsions sur une sortie du port P1 :

```
                MOV     R2,#8
REPET :         CPL     P1.7
                DJNZ    R2,REPET
```

Formes possibles :

DJNZ Rn, rel	2 octets	Cycle : 2
DJNZ direct, rel	3 octets	Cycle : 2

INC <byte>

Fonction : Incrémente la variable byte

Description : La variable spécifiée comme opérande est incrémentée de 1. Une valeur d'origine de 0FFH se traduira par le débordement vers 00H. Aucun indicateur n'est modifié.

Nota : Quand cette instruction est utilisée pour modifier une sortie de port, la valeur utilisée comme origine sera lue à partir du tampon de sortie de ce port et non pas sur les broches d'entrée.

Exemple : Le registre 0 contient la valeur 7EH. Les mémoires internes d'adresses 7EH et 7FH contiennent respectivement 0FFH et 40H. La séquence ...

```
INC @R0
INC R0
INC @R0
```

donne au registre R0 la valeur 7FH et aux deux mémoires 7EH et 7FH les valeurs 00H et 41H.

Formes possibles :

INC A	1 octet	Cycle : 1
INC Rn	1 octet	Cycle : 1
INC direct	2 octets	Cycle : 1
INC @ Ri	1 octet	Cycle : 1

INC DPTR

Fonction : Incrémente le pointeur de données

Description : Incrémente de 1 le pointeur 16 bits formé de DPH et DPL. Les indicateurs ne sont pas modifiés par cette instruction. C'est le seul registre de 16 bits qui soit concerné par cette instruction.

Exemple : Les registres DPH et DPL contiennent 12H et 0FEH respectivement. La séquence...

```
INC DPTR

INC DPTR

INC DPTR
```

change DPH et DPL en 13H et 01H

Format : 1

Cycle : 2

JB bit, rel

Fonction : Saut si le bit est à 1

Description : Si le bit indiqué est à 1, un saut relatif est exécuté. Dans le cas contraire, c'est l'instruction suivante qui est exécutée. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci. Le bit testé n'est pas modifié. Aucun indicateur n'est modifié.

Exemple : La donnée présente sur le port 1 est 11001010B. L'accumulateur contient la valeur 56H (01010110B). La séquence ...

```
JB P1.2,ET1

JB ACC.2,ET2
```

provoque un branchement à l'adresse ET2.

Format : 3 octets

Cycle : 2

JBC bit, rel

Fonction : Saut si le bit est à 1 et mise à 0 de celui-ci.

Description : Si le bit indiqué est à 1, un saut à l'adresse indiquée est exécuté. Dans le cas contraire, c'est l'instruction suivante qui est exécutée. Si le test est positif le bit est remis à 0. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci. Aucun indicateur n'est modifié.

Nota : Quand cette instruction est utilisée pour modifier une sortie de port, la valeur utilisée comme origine sera lue à partir du tampon de sortie de ce port et non pas sur les broches d'entrée.

Exemple : L'accumulateur contient la valeur 56H (01010110B). La séquence

```
JBC ACC.3,ET1
JBC ACC.2,ET2
```

provoque un branchement à l'adresse ET2 avec l'accumulateur égal à 52H (01010010B).

Format : 3 octets

Cycle : 2

JC rel

Fonction : Saut si indicateur de retenue à 1

Description : Si l' indicateur de retenue est à 1, un saut à l'adresse spécifiée est exécuté. Sinon c'est l'instruction suivante qui est exécutée. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci. Aucun indicateur n'est modifié.

Exemple : L'indicateur de retenue est à 0. La séquence ...

```
JC ET1
CPL C
JC ET2
```

positionnera l'indicateur de retenue à 1 puis réalisera un branchement à l'adresse ET2.

Format : 2 octets

Cycle : 2

JMP @A+ DPTR

Fonction : Saut indirect

Description : La valeur 8 bits non signée contenue dans l'accumulateur est additionnée à la valeur 16 bits contenue dans DPTR. Le résultat constitue l'adresse de la prochaine instruction qui sera exécutée.

Exemple : Une valeur paire comprise entre 0 et 6 est placée dans l'accumulateur. La séquence suivante réalisera un saut à partir de la table SAUT

```
        MOV     DPTR,#SAUT
        JMP     @A+DPTR
SAUT:   AJMP    ET0
        AJMP    ET1
        AJMP    ET2
        AJMP    ET3
```

si l'accumulateur valait 4 au début de la séquence, c'est le saut vers l'adresse ET2 qui a été exécuté. Il faut se souvenir qu'une instruction AJMP n'occupe que 2 octets.

Format : 1 octet

Cycle : 2

JNB bit, rel

Fonction : Saut si bit non à 1 (saut si Bit = 0)

Description : Si le bit spécifié est à 0, un saut relatif est réalisé, sinon l'instruction suivante est normalement exécutée. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci.

Exemple : La donnée présente sur les entrées du port 1 est 11001010 B. L'accumulateur vaut 56H (01010110 B), la séquence

```
JNB     P1.3,ET1
JNB     ACC.3,ET2
```

provoque un saut vers l'adresse ET2, car le bit 3 de l'accumulateur est à 0.

Format : 3 octets

Cycle : 2

JNC rel

Fonction : Saut si l'indicateur de retenue est à 0

Description : Si l'indicateur de retenue est à 0, un branchement à l'adresse indiquée est exécuté. Dans le cas contraire, c'est l'instruction suivante qui sera exécutée. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci. Les indicateurs ne sont pas modifiés.

Exemple : L'indicateur de retenue est à 1. La séquence suivante

```
JNC     ET1
CPL     C
JNC     ET2
```

remet l'indicateur de retenue à 0 et provoque un branchement à l'adresse ET2.

Format : 2 octets

Cycle : 2

JNZ rel

Fonction : Saut si accumulateur différent de 0

Description : Si l'accumulateur contient une valeur différente de 0, un branchement à l'adresse spécifiée est exécuté. Dans le cas contraire, c'est l'instruction suivante qui sera exécutée. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci.

Exemple : L'accumulateur vaut 00H. La séquence suivante ...

```
JNZ     ET1
INC     A
JNZ     ET2
```

donne à l'accumulateur la valeur 01H et provoque un branchement à l'adresse ET2

Format : 2 octets

Cycle : 2

JZ rel

Fonction : Saut si accumulateur égal 0

Description : Si tous les bits de l'accumulateur sont à 0, le branchement à l'adresse spécifiée est réalisé. Dans le cas contraire, c'est l'instruction suivante qui sera exécutée. L'adresse de destination est calculée par addition de la valeur signée "rel" à la valeur du PC après incrémentation de celle-ci. Aucun registre n'est modifié.

Exemple : L'accumulateur vaut 01H. La séquence suivante :

```
JZ      ET1
DEC     A
JZ      ET2
```

donne à l'accumulateur la valeur 0 et provoque un branchement à l'adresse ET2.

Format : 2 octets

Cycle : 2

LCALL addr16

Fonction : Appel d'un sous-programme

Description : Provoque un branchement à l'adresse exprimée par 16 bits. L'instruction additionne la valeur 3 au contenu du PC afin de constituer l'adresse de la prochaine instruction à exécuter. Celle-ci est alors sauvegardée dans la pile. Le pointeur de pile est donc incrémenté de 2 par cette opération. Puis le branchement à l'adresse du sous-programme est réalisé. Aucun registre n'est modifié.

Exemple : Le pointeur de pile est égal à 07H. L'étiquette "SOUSPG" correspond à l'adresse 1234H. Après exécution de l'instruction

```
LCALL  SOUSPG
```

située à l'adresse 0123H, le pointeur de pile vaut 09H, le contenu des mémoires d'adresse 08H et 09H est respectivement 26H et 01H et le PC vaut 1235H.

Format : 3 octets

Cycle : 2

LJMP addr16

Fonction : Saut inconditionnel long

Description : Cette instruction réalise un saut inconditionnel à une adresse spécifiée par 16 bits. La destination peut se situer à n'importe quel endroit des 64Ko de la mémoire programme. Aucun registre n'est modifié.

Exemple : L'étiquette "SAUTADR" correspond à l'adresse 1234H. L'instruction

LJMP SAUTADR

provoque un branchement à l'adresse 1234H.

Format : 3 octets

Cycle : 2

MOV <dest. byte>, <src. byte>

Fonction : Transfert.

Description : L'opérande source est copié dans l'opérande destination. L'opérande source n'est pas modifié. Aucun indicateur n'est modifié. C'est une des instructions les plus souples. 15 combinaisons d'adressage source et destination sont permises.

Exemple : La mémoire interne d'adresse 30H contient la valeur 40H. La valeur de la mémoire interne d'adresse 40H est 10H. La donnée présente sur le port P1 est 11001010 B (0CAH).

```
MOV     R0,#30H     ;R0 ← 30H
MOV     A,@R0       ;A ← 40H
MOV     R1,A        ;R1 ← 40H
MOV     B ,@R1      ;B ← 10H
MOV     @R1,P1      ;RAM(40H) ← 0CAH
MOV     P2,P1       ;P2 ← 0CAH
```

donne la valeur 30H au registre R0, la valeur 40H à l'accumulateur et au registre R1, la valeur 10H au registre B, et la valeur 0CAH à la mémoire d'adresse 40H et au port P2.

Formes possibles :

MOV A,Rn	1 octet	Cycle : 1
MOV A,direct	2 octets	Cycle : 1
MOV A, @Ri	1 octet	Cycle : 1
MOV A, #data	2 octets	Cycle : 1
MOV Rn, A	1 octet	Cycle : 1
MOV Rn, direct	2 octets	Cycle : 2
MOV Rn, #data	2 octets	Cycle : 1
MOV direct, A	2 octets	Cycle : 1
MOV direct, Rn	2 octets	Cycle : 2
MOV direct, direct	3 octets	Cycle : 2
MOV direct, @ Ri	2 octets	Cycle : 2
MOV direct, # data	3 octets	Cycle : 2
MOV @ Ri, A	1 octet	Cycle : 1
MOV @ Ri, direct	2 octets	Cycle : 2
MOV @ Ri, # data	2 octets	Cycle : 1

MOV <dest.bit>, <src.bit>

Fonction : Transfert au niveau du bit.

Description : L'état logique du bit source est copié vers le bit destination. Un des deux bits opérandes doit être l'indicateur de retenue, et l'autre peut être un des bits directement adressables. Aucun indicateur n'est modifié.

Exemple : L'indicateur de retenue est à 1. La donnée présente sur le port 3 est 11000101 B. La donnée écrite précédemment sur la sortie du port 1 est 00110101 B.

```
MOV     P1.3,C
MOV     C,P3.3
MOV     P1.2,C
```

positionne l'indicateur de retenue à 0 et change le port 1 en 00111001 B.

Formes possibles :

MOV C, bit	2 octets	Cycle : 1
MOV bit, C	2 octets	Cycle : 2

MOV DPTR, #data16

Fonction : Chargement du pointeur 16 bits avec une constante

Description : Le pointeur de données DPTR est chargé de la valeur 16 bits qui suit. Le second octet de l'instruction est le poids fort de la donnée. Il est placé dans DPH, quant au troisième octet, le poids faible de la donnée, il est placé dans DPL. Aucun indicateur n'est modifié. C'est la seule instruction de chargement 16 bits du jeu d'instructions.

Exemple : L'instruction

MOV DPTR ,#1234H

chargera la valeur 1234H dans DPTR avec 12H dans DPH et 34H dans DPL.

Format : 3 octets

Cycle : 2

MOVC A, @A+<base-reg>

Fonction : Transfert d'un octet depuis la mémoire programme.

Description : L'instruction MOVC charge l'accumulateur avec un code ou une constante situé dans la mémoire programme. L'adresse de l'octet recherché est égale à la somme de la valeur 8 bits d'origine de l'accumulateur et du contenu de 16 bits du registre de base. Le registre de base peut être le compteur programme PC ou le registre DPTR. Si le registre de base est le PC, celui-ci est d'abord incrémenté pour pointer l'instruction suivante. Aucun indicateur n'est modifié.

Exemple : Une valeur entre 0 et 3 est placée dans A. La suite d'instructions, ci-dessous, incrémente d'abord de 1 l'accumulateur pour compenser l'octet occupé par l'instruction "RET" et retourne dans A l'une des 4 valeurs définies par la directive d'assemblage DB.

```
INC     A
MOVC    A,@A+PC
RET
DB      66H
DB      77H
DB      88H
DB      99H
```

Si ce sous-programme est appelé avec l'accumulateur égal à 01, en retour l'accumulateur vaudra 77H.

Formes possibles :

MOVC A, @A+DPTR	1 octet	Cycle : 2
MOVC A, @A+PC	1 octet	Cycle : 2

MOVX <dest.byte>, <src.byte>

Fonction : Transfert externe.

Description : L'instruction MOVX permet un transfert entre l'accumulateur et la mémoire externe. Il existe deux types d'instructions caractérisés par un champ d'adressage différent de la mémoire externe : adressage indirect sur 8 bits ou sur 16 bits. Dans le premier type d'adressage, le contenu de R0 ou de R1, de la banque de registre active, constitue l'adresse 8 bits qui est multiplexée à la donnée sur le port 0. Si cet espace d'adressage sur 8 bits était insuffisant, il est toujours possible d'utiliser un second port dont l'état serait contrôlé par une instruction de sortie vers les broches de ce port. Dans le second type d'adressage, le pointeur de données DPTR contient une adresse de 16 bits. Le port P2 se voit affecter du contenu de DPH alors que le port P0 multiplexe les 8 bits de poids faible de l'adresse (DPL) avec la donnée. Il faut noter que le contenu du registre de fonction P2 conserve la valeur qu'il avait avant ce type d'adressage et que seul le tampon de sortie du port 2 est modifié par la partie haute de l'adresse, le temps de la lecture ou de l'écriture de la mémoire externe.

Exemple : Un espace mémoire externe de 256 octets est connecté au port P0. Adresses et données sont multiplexées sur ce port. Un des bits du port P3 est utilisé pour valider le circuit mémoire. Les registres R0 et R1 contiennent respectivement les valeurs 12H et 34H. La mémoire externe d'adresse 34H contient la valeur 56H. La séquence

```
MOVX    A,@R1
MOVX    @R0,A
```

copie la valeur 56H successivement dans l'accumulateur et dans la mémoire externe d'adresse 12H.

Formes possibles :

MOVX A, @Ri	1 octet	Cycle : 2
MOVX A, @DPTR	1 octet	Cycle : 2
MOVX @Ri, A	1 octet	Cycle : 2
MOVX @DPTR, A	1 octet	Cycle : 2

MUL AB

Fonction : Multiplication.

Description : MUL AB réalise la multiplication non signée des entiers contenus dans l'accumulateur et le registre B. Le poids faible du résultat 16 bits est placé dans l'accumulateur, alors que les 8 bits de poids fort sont placés dans le registre B. Si le produit donne un résultat supérieur à 255, l'indicateur de débordement OV est mis à 1. Dans le cas contraire il est mis à 0. L'indicateur de retenue est dans tous les cas forcé à 0.

Exemple : Au départ l'accumulateur contient la valeur 80 (50H). Le registre B contient la valeur 160 (0A0H). L'instruction

```
MUL     AB
```

donne le produit 12800 (3200H). B prend donc la valeur 32H et A, la valeur 00 . l'indicateur OV est mis à 1.

Format : 1 octet

Cycle : 4

NOP

Fonction : Pas d'opération.

Description : L'exécution continue à la prochaine instruction. A part le PC, aucun registre ou indicateur n'est modifié.

Exemple : La suite d'instructions

```
CLR   P2.7
NOP
NOP
NOP
NOP
SETB  P2.7
```

permet de produire une courte impulsion négative sur la ligne 7 du port P2. La suite CLR/SETB provoque une impulsion négative correspondant à un cycle machine. En ajoutant des instructions "NOP" on peut augmenter la durée de cette impulsion par pas d'un cycle machine. Ici, l'impulsion dure 5 cycles machine.

Format : 1 octet (00H)

Cycle : 1

ORL <dest. byte>, <src. byte>

Fonction : OU logique.

Description : Un OU logique est réalisé entre l'octet source et l'octet destination. Le résultat est rangé dans l'opérande destination. Aucun indicateur n'est modifié. Les deux opérandes peuvent représenter 6 combinaisons de mode d'adressage. Lorsque l'accumulateur est considéré comme opérande destination, l'opérande source peut être un registre, une adresse directe, un adressage indirect par registre, ou une donnée immédiate ; quand l'opérande source est une adresse directe, l'opérande source peut être l'accumulateur ou une donnée immédiate. Remarque : Quand ce type d'instruction est utilisé pour modifier l'état d'un port, la valeur utilisée comme état d'origine du port sera lue à partir des sorties du registre du port et non pas à partir de l'état des broches du circuit intégré.

Exemple : L'accumulateur contient la valeur 0C3H (11000011B) et R0 contient la valeur 55H (01010101B). Après l'exécution de l'instruction

```
ORL A,R0
```

l'accumulateur recevra la valeur 0D7H (11010111B).

Formes possibles :

ORL A, Rn	1 octet	Cycle : 1
ORL A, direct	2 octets	Cycle : 1
ORL A, @ Ri	1 octet	Cycle : 1
ORL A, # data	2 octets	Cycle : 1
ORL direct, A	2 octets	Cycle : 1
ORL direct, #data	3 octets	Cycle : 2

ORL C, <src.bit>

Fonction : OU logique au niveau du bit

Description : Positionne à 1 l'indicateur de retenue si le bit source est à 1. L'indicateur de retenue est utilisé comme "accumulateur" par le processeur logique. Le signe / qui précède éventuellement le bit opérande indique que l'opération OU doit porter sur le complément de l'état de ce bit. Le bit source n'est pas modifié par cette opération.

Exemple : Positionner à 1 l'indicateur de retenue C, si P1.0=1, ou si ACC.7=1, ou si OV=0

```
MOV C,P1.0
charger indicateur de retenue avec P1.0

ORL C,ACC.7
ou logique avec ACC.7

ORL C,/OV
ou logique avec complément de OV
```

Formes possibles :

ORL C, bit	2 octets	Cycle : 2
ORL C, /bit	2 octets	Cycle : 2

POP direct

Fonction : Récupération depuis la pile.

Description : Le contenu de la RAM interne, adressée par le pointeur de pile, est lu puis le pointeur de pile est incrémenté de 1. La valeur lue est copiée à l'adresse indiquée par l'opérande de l'instruction. Aucun indicateur n'est modifié.

Exemple : Le pointeur de pile contient l'adresse de valeur 32H, et les mémoires d'adresse 30H, 31H, 32H contiennent respectivement les valeurs 20H, 23H, et 01H. La séquence suivante

```
POP DPH
POP DPL
```

positionne le pointeur de pile sur l'adresse 30H et le pointeur de données DPTR a pris la valeur 0123H. A ce stade, l'instruction

```
POP SP
```

positionne le pointeur de pile sur l'adresse 20H. Il faut noter que dans ce cas, le pointeur a été décrémenté de 1 (2FH) avant d'être affecté de la valeur 20H.

Format : 2 octets

Cycle : 2

PUSH direct

Fonction : Sauvegarde sur la pile.

Description : Le pointeur de pile est incrémenté de 1. Le contenu de l'adresse indiquée est ensuite copié dans la RAM interne pointée par SP. Aucun indicateur n'est modifié par cette opération.

Exemple : Le pointeur SP contient la valeur 09H, et le pointeur de données DPTR vaut 0123H. La séquence

```
PUSH DPL
PUSH DPH
```

déplace le pointeur SP à la valeur 0BH et place en mémoire RAM interne les valeurs 23H et 01H aux adresses 0AH et 0BH.

Format : 2 octets

Cycle : 2

RET

Fonction : Retour de sous-programme.

Description : L'instruction RET assure la récupération depuis la pile, de la valeur du compteur programme PC et l'exécution continue à partir de cette adresse récupérée. Aucun indicateur n'est modifié par cette opération.

Exemple : Le pointeur de pile contient la valeur 0BH. Les RAM internes d'adresse 0AH et 0BH contiennent respectivement les valeurs 23H et 01H. L'instruction

```
RET
```

donne au pointeur de pile la valeur 09H et l'exécution se poursuit à l'adresse 0123H.

Format : 1 octet

Cycle : 2

RETI

Fonction : Retour d'interruption.

Description : L'instruction RETI assure la récupération depuis la pile, de la valeur du compteur programme PC. La logique de niveau d'interruption est remise à jour. Le pointeur de pile est décrémenté de 2. Aucun indicateur n'est modifié par cette opération. L'exécution se poursuit à l'adresse récupérée depuis la pile.

Exemple : Le pointeur de pile contient la valeur 0BH. Les RAM internes d'adresse 0AH et 0BH contiennent respectivement les valeurs 23H et 01H. L'instruction

RETI

donne au pointeur de pile la valeur 09H et l'exécution se poursuit à l'adresse 0123H.

Format : 1 octet

Cycle : 2

RL A

Fonction : Rotation vers gauche.

Description : Les 8 bits de l'accumulateur sont décalés d'un cran vers la gauche. Le bit B7 est placé en poids faible (B0). Aucun indicateur n'est modifié par cette opération.

Exemple : L'accumulateur contient la valeur 8BH (10001011B). L'instruction

RL A

donne à l'accumulateur la valeur 17H (00010111B).

Format : 1 octet

Cycle : 1

RLC A

Fonction : rotation vers gauche à travers l'indicateur de retenue.

Description : Les 8 bits de l'accumulateur sont décalés d'un cran vers la gauche. Le bit B7 est placé dans l'indicateur de retenue, alors que la valeur de celui-ci est placée en poids faible de l'accumulateur.

Exemple : L'accumulateur vaut 0C5H (11000101B), et l'indicateur de retenue vaut 0. L'instruction

RLC A

donne à l'accumulateur la valeur 8AH (10001010B), et la valeur 1 à l'indicateur de retenue.

Format : 1 octet

Cycle : 1

RR A

Fonction : Rotation vers droite.

Description : Les 8 bits de l'accumulateur sont décalés d'un rang vers la droite. Le bit B0 est placé en pois fort (B7). Aucun indicateur n'est modifié par cette opération.

Exemple : L'accumulateur contient la valeur 8BH (10001011B). L'instruction

RR A

donne à l'accumulateur la valeur C5H (11000101B).

Format : 1 octet

Cycle : 1

RRC A

Fonction : Rotation vers droite à travers l'indicateur de retenue.

Description : Les 8 bits de l'accumulateur et l'indicateur de retenue sont décalés ensemble d'un rang vers la droite. B0 est placé dans l'indicateur de retenue alors que la valeur de celui-ci est placée en B7.

Exemple : L'accumulateur vaut 0C5H (11000101B), et l'indicateur de retenue vaut 0. L'instruction

```
RRC A
```

donne à l'accumulateur la valeur 62 H (01100010B), et la valeur 1 à l'indicateur de retenue.

Format : 1 octet

Cycle : 1

SETB <bit>

Fonction : Mise à 1 d'un bit.

Description : Les instructions SETB positionnent à la valeur 1, le bit spécifié par l'opérande.

Exemple : L'indicateur de retenue est à 0. Le port P1 vaut 34H (00110100B). La séquence

```
SETB C
SETB P1.0
```

positionne l'indicateur de retenue à la valeur 1 et donne au port P1 la valeur 35H (00110101B).

Formes possibles :

SETB C	1 octet	Cycle : 1
SETB bit	2 octets	Cycle : 1

SJMP rel

Fonction : Saut.

Description : Cette instruction réalise un branchement inconditionnel à l'adresse indiquée. L'adresse de destination est calculée par addition du déplacement signé, exprimé sur le deuxième octet de l'instruction, à la valeur du PC après incrémentation de deux fois. L'espace de saut est ainsi limité à 128 octets en amont, et 127 octets en aval de cette instruction.

Exemple : L'étiquette ADRREL correspond à l'adresse 0123H. L'instruction

```
SJMP    ADRREL
```

provoque un branchement à l'adresse 0123H.

Format : 2 octets

Cycle : 2

SUBB A, <src.byte>

Fonction : soustraction avec retenue.

Description : L'opérande source et l'indicateur de retenue sont soustraits du contenu de l'accumulateur. Le résultat est rangé dans l'accumulateur. L'indicateur de retenue est mis à 1 si un débordement survient. Ce mécanisme permet une soustraction en multiple précision où l'on soustrait l'indicateur de retenue résultant de l'opération précédente. L'indicateur AC est positionné si un report est provoqué par le bit 3. L'indicateur OV est positionné si un report est provoqué par le bit 6 mais pas par le bit 7 et inversement.

Dans une soustraction d'entiers signés, l'indicateur OV est mis à 1 :

si une valeur négative soustraite d'une valeur positive provoque un résultat négatif ;

si une valeur positive soustraite d'une valeur négative provoque un résultat positif.

Exemple : L'accumulateur vaut 0C9H et R2 vaut 54H avec l'indicateur de retenue à 1. l'instruction

SUBB A,R2

place la valeur 74H dans l'accumulateur. L'indicateur de retenue et AC sont mis à 0 et OV est positionné à 1.

Formes possibles :

Formes possibles :

SUBB A, Rn	1 octet	Cycle : 1
SUBB A, direct	2 octets	Cycle : 1
SUBB A, @ Ri	1 octet	Cycle : 1
SUBB A, # data	2 octets	Cycle : 1

SWAP A

Fonction : Permutation de quartets dans l'accumulateur.

Description : L'instruction "SWAP A" échange les 4 bits de poids fort avec les 4 bits de poids faible de l'accumulateur. Cette instruction remplace avantageusement 4 instructions de rotation.

Exemple : L'accumulateur contient la valeur 0C5H (11000101B). L'instruction

SWAP A

donne à l'accumulateur, la valeur 5CH (01011100B).

Format : 1 octet

Cycle : 1

XCH A, <byte>

Fonction : Échange de contenus.

Description : L'instruction XCH charge l'accumulateur avec le contenu de l'opérande source et dans le même temps, la valeur d'origine de l'accumulateur est copiée dans l'opérande source.

Exemple : R0 contient l'adresse de valeur 20H. l'accumulateur contient la valeur 3FH. La RAM interne d'adresse 20H contient la donnée 75H. L'instruction

XCH A,@R0

charge la valeur 3FH dans la RAM d'adresse 2OH et donne à l'accumulateur la valeur 75H.

Formes possibles :

XCH A, Rn	1 octet	Cycle : 1
XCH A, direct	2 octets	Cycle : 1
XCH A, @ Ri	1 octet	Cycle : 1

XCHD A, @Ri

Fonction : Échange d'un quartet.

Description : XCHD échange le quartet de poids faible (B3/B0) de l'accumulateur avec celui de la RAM interne adressée indirectement par le registre R0 ou R1. Les quartets de poids fort ne sont ni échangés, ni modifiés. Aucun indicateur n'est modifié par cette opération.

Exemple : R0 contient l'adresse de valeur 20H. L'accumulateur contient la valeur 36H. La RAM interne d'adresse 20H contient la valeur 75H. L'instruction

XCHD A,@R0

donne la valeur 76H à la RAM interne dont l'adresse est 20H et l'accumulateur contient à présent la valeur 35H.

Format : 1 octet

Cycle : 1

XRL <dest.byte>, <src.byte>

Fonction : OU exclusif logique.

Description : Un OU exclusif est réalisé entre l'octet source et l'octet destination. Le résultat est rangé dans l'opérande destination. Aucun indicateur n'est modifié. Les deux opérandes peuvent représenter 6 combinaisons de mode d'adressage. Lorsque l'accumulateur est considéré comme opérande destination, l'opérande source peut être un registre, une adresse directe, un adressage indirect par registre, ou une donnée immédiate ; quand l'opérande source est une adresse directe, l'opérande source peut être l'accumulateur ou une donnée immédiate.

Remarque : Quand ce type d'instruction est utilisé pour modifier l'état d'un port, la valeur utilisée comme état d'origine du port sera lue à partir des sorties du verrou du port et non pas à partir de l'état des broches du port.

Exemple : L'accumulateur contient la valeur 0C3H (11000011B) et R0 contient la valeur 0AAH (10101010B). Après l'exécution de l'instruction

```
XRL A,R0
```

l'accumulateur recevra la valeur 69H (01101001B)

Quand l'opérande destination est une adresse directe, cette instruction peut permettre un complément logique des bits de la RAM interne ou des registres matériels. Les bits à complémenter sont déterminés par un octet de masque qui peut être une constante dans le programme ou le résultat d'un calcul réalisé à partir de l'accumulateur. L'instruction

```
XRL P1, #00110001B
```

complémente les bits B0, B4, et B5 du port P1.

Formes possibles :

XRL A, Rn	1 octet	Cycle : 1
XRL A, direct	2 octets	Cycle : 1
XRL A, @ Ri	1 octet	Cycle : 1
XRL A, # data	2 octets	Cycle : 1
XRL direct, A	2 octets	Cycle : 1
XRL direct,#data	3 octets	Cycle : 2

8

Détail de la mémoire vive interne

Une des différences majeures entre les versions 8051 et 8052 réside dans la taille mémoire RAM interne. Elle est constituée de 128 octets pour le 8051 et de 256 octets pour le 8052, auxquels viennent se rajouter, dans les deux cas, les 128 octets du bloc de registres à fonction spéciale.

Organisation générale de la RAM interne

Cette mémoire vive occupe l'espace d'adresses 00 à 7FH ou 00 à 0FFH. L'espace commun aux deux versions (00 à 7FH) comporte des zones dotées de propriétés particulières qui peuvent être exploitées si l'application le nécessite. Dans le cas contraire ces octets de mémoire RAM peuvent être utilisés normalement. Le tableau 8.1 précise les particularités de cette zone mémoire.

Une mémoire externe de données peut malgré tout occuper le même espace ; l'accès à ce double plan de mémoire est rendu possible par restriction des modes d'adressage utilisables au niveau logiciel.

Un troisième plan mémoire existe dans cet espace d'adressès. Il est occupé par les **registres à fonction spéciale** (SFR). Les registres à fonction spéciale ne sont accessibles, au niveau logiciel, qu'en adressage **direct**. La mémoire RAM interne, éventuellement superposée (seulement sur le 8052), est adressable en mode **indirect par registre**.

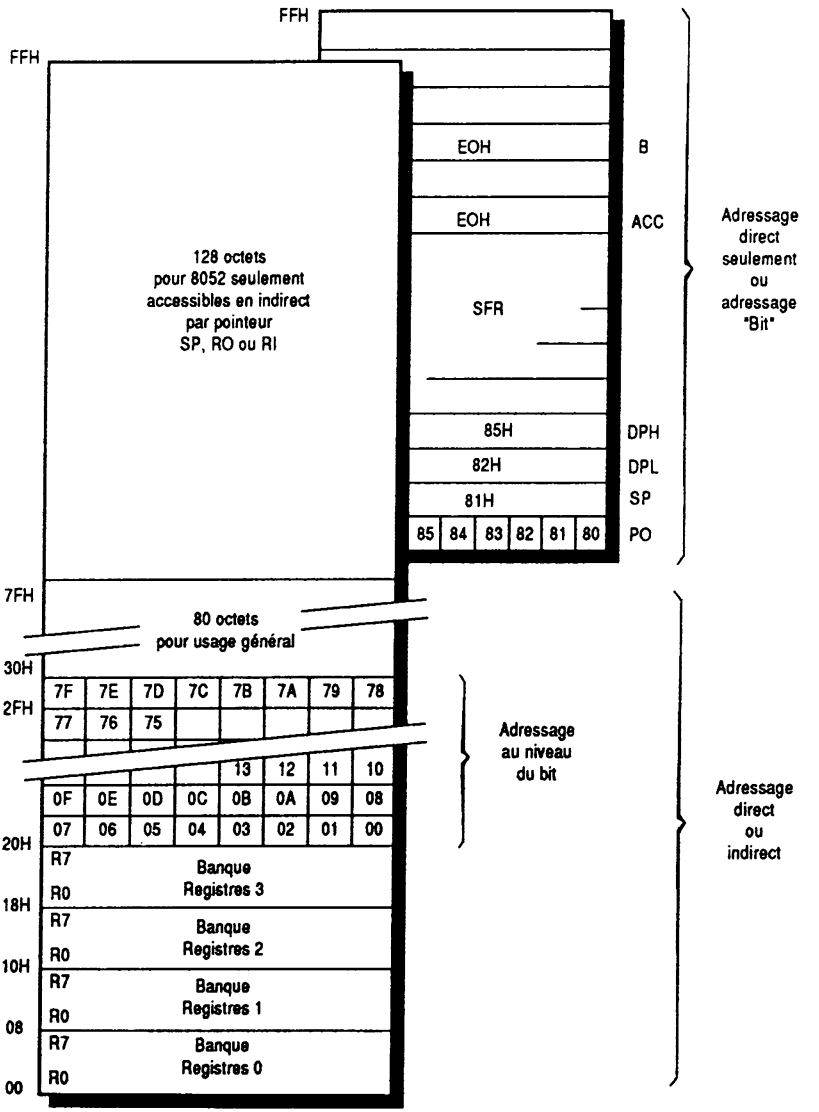


Tableau 8.1 : Organisation et particularités de la RAM interne

Ainsi

```
MOV  A,99H  (adressage direct)
```

place dans l'accumulateur le contenu du SFR situé à l'adresse 99H alors que

```
MOV  R0,#99H  ;R0=99H
MOV  A, @R0   ; adressage indirect par registre R0
```

place dans l'accumulateur le contenu de la RAM dont l'adresse est 99H et que

```
MOV  R0,#99H  ;R0=99H
MOVX A, @R0
```

place dans A le contenu de la mémoire externe de données d'adresse 99H.

Registres universels

Les 32 premiers octets de la RAM interne peuvent être utilisés comme des registres universels. En fait, seulement 8 de ces 32 octets peuvent être activés à la fois comme registre. On parle alors de 4 banques de 8 registres. L'activation d'une banque est possible à tout instant par programmation. Le fait d'activer une banque désactive les autres. Au niveau de la programmation la dénomination d'un registre (par exemple R0) ne désignera donc pas obligatoirement la même mémoire physique. Ainsi R0 peut désigner les mémoires d'adresse 0, 08, 10H et 18H selon que l'on a activé la banque 0, 1, 2 ou 3. La sélection de banque s'effectue par positionnement des deux bits RS0 et RS1 du registre PSW. Un changement de banque de registres peut s'avérer un moyen rapide et pratique de sauvegarder un contexte lors d'un traitement d'interruption. Par défaut (à l'initialisation) c'est la banque 0 (adresses 00 à 07) qui est active. Les registres R sont à usage universel bien que seuls les registres R0 et R1 de chaque banque permettent l'adressage indirect par registre.

adresse		Banque n°	RS1	RS0
1FH	R7			
18H	R0	3	1	1
17H	R7			
10H	R0	2	1	0
0FH	R7			
08H	R0	1	0	1
07H	R7			
00H	R0	0	0	0

Tableau 8.2 : Zone RAM réservée aux 4 banques de registres

Zone mémoire adressable au niveau du bit

De l'adresse 20H à l'adresse 2FH se trouvent 16 octets qui ont la particularité d'être adressables au niveau du bit. L'adresse de chacun des bits est exprimée sur un octet. Ainsi le bit de poids faible de l'octet d'adresse 20H, peut être adressé directement par l'adresse 00, le bit de poids fort de ce même octet porte l'adresse 07 et le bit de poids fort de l'octet d'adresse 2FH porte l'adresse 7FH (voir tableau 8.3).

Il est à remarquer que certains registres SFR sont eux aussi adressables au niveau du bit (voir tableau 8.4).

adresse	b7	b6	b5	b4	b3	b2	b1	b0
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00

Tableau 8.3 : Adresses des bits de la zone adressable au niveau du bit

Registres à fonction spéciale (SFR)

Les registres à fonction spéciale occupent les adresses directes 80H à 0FFH. Le détail de l'implantation de ces registres est donné dans le tableau 8.4.

Symbole	Fonction	Adresse directe	Adresse au niveau du bit, symbole ou fonction secondaire								Etat initial
B	reg. pour multiplication et division	F0H *	F7	F6	F5	F4	F3	F2	F1	F0	00000000
ACC	Accumulateur	E0H *	E7	E6	E5	E4	E3	E2	E1	E0	00000000
PSW	Mot d'état programme	D0H *	D7	D6	D5	D4	D3	D2	D1	D0	00000000
			CY	AC	F0	RS1	RS0	OV	–	P	
TH2	# Poids fort du Timer 2	CEH									00000000
TL2	# poids faible du Timer 2	CDH									00000000
RCAP2H	# Capture/rechargement Timer 2	CCH									00000000
RCAP2L	# Capture/rechargement Timer 2	CBH									00000000
T2CON	# Contrôle du Timer 2	C8H *	CF	CE	CD	CC	CB	CA	C9	C8	00000000
			TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	
IP	Niveau de priorité pour interruptions	B8H *	BF	BE	BD	BC	BB	BA	B9	B8	XX000000
			–	–	PT2	PS	PT1	PX1	PT0	PX0	
P3	PORT P3	B0H *	B7	B6	B5	B4	B3	B2	B1	B0	11111111
			RD	WR	T1	T0	INT1	INT0	TxD	RxD	
IE	Validation des interruptions	A8H *	AF	AE	AD	AC	AB	AA	A9	A8	0X000000
			EA	–	ET2	ES	ET1	EX1	ET0	EX0	
P2	PORT P2	A0H *	A7	A6	A5	A4	A3	A2	A1	A0	11111111
			A15	A14	A13	A12	A11	A10	A9	A8	
SBUF	Registre de données du port série	99H									00000000
SCON	Contrôle du port série	98H *	9F	9E	9D	9C	9B	9A	99	98	00000000
			SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
P1	PORT P1	90H *	97	96	95	94	93	92	91	90	11111111
			–	–	–	–	–	–	T2EX	T2	
TH1	Poids fort du Timer 1	8DH									00000000
TH0	Poids fort du Timer 0	8CH									00000000
TL1	Poids faible du Timer 1	8BH									00000000
TL0	poids faible du Timer 0	8AH									00000000
TMOD	Modes pour Timers 0 & 1	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00000000
TCON	Contrôle des Timers 0 & 1	88H *	8F	8E	8D	8C	8B	8A	89	88	00000000
			TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
PCON	Mode de consommation pour CMOS	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxx
DPH	Poids fort du pointeur DPTR	83H									00000000
DPL	Poids faible du pointeur DPTR	82H									00000000
SP	Pointeur de PILE	81H									00000111
P0	Registre du Port P0	80H *	87	86	85	84	83	82	81	80	11111111
			AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	

Registres à fonction spéciale (SFR)

* adressable au niveau du bit

\# seulement dans les versions 8052

Tableau 8.4 : Registres à fonction spéciale

Accumulateur

ACC représente l'accumulateur. La mnémonique utilisée dans le jeu d'instructions est simplement A. Le registre ACC peut être considéré comme une image de A. ACC est adressable au niveau du bit. L'instruction MOV A, ACC qui signifierait copier l'accumulateur dans l'accumulateur n'est pas une instruction valide.

Registre B

Le registre B est dédié aux opérations de multiplication et de division. Pour les autres opérations il peut être utilisé mais son utilisation nécessite l'adressage direct. L'assembleur MCS_51 reconnaît le symbole B et le remplace par son adresse 0F0H.

Exemple :

```
MOV  B,  #5   ; placer la donnée 5 à l'adresse 0F0H
MOV  A,  #6   ; placer la donnée 6 dans l'accumulateur
MUL  AB       ; multiplier A par B
```

Registre d'état PSW

PSW est l'abréviation de Program Status Word (mot d'état programme). Ce registre contient les indicateurs d'état. Ces indicateurs peuvent être lus ou modifiés par l'utilisateur.

PSW : Adresse directe DOH et adressable au niveau BIT

D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	—	P

CY : indicateur de retenue (désigné aussi par la lettre C)
AC : indicateur de retenue auxiliaire indiquant un report du bit 3 vers le bit 4 (utilisé pour les opérations en code BCD).
FO : indicateur sans fonction, mis à disposition de l'utilisateur.
RS1 : Sélection d'une banque de registres.
RS0 : Sélection d'une banque de registres.

RS1	RS0	banque active	adresse
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

OV : indicateur de débordement
P : indicateur de parité.

Pointeur de pile SP

Le registre SP est un pointeur de pile au format 8 bits. Cette caractéristique limite théoriquement la taille de la pile à 256 octets. Pratiquement, la pile fera moins de 256 octets car celle-ci doit forcément se situer dans la mémoire vive interne. SP est incrémenté avant la sauvegarde d'une donnée dans la pile. Après initialisation du microcontrôleur, ce pointeur de pile a par défaut la valeur 07, ce qui place la pile juste au-dessus de la banque de registres. Cette valeur peut, bien sûr, être redéfinie par l'utilisateur et la pile peut résider à n'importe quelle adresse de la RAM INTERNE.

Pointeur de données DPTR

Il est constitué d'un registre poids fort (DPH) et d'un registre poids faible (DPL). Sa fonction normale est de pointer une donnée située dans la mémoire externe de données dont l'adresse est égale à la valeur 16 bits contenue dans DPTR. Les deux registres DPL et DPH peuvent être éventuellement modifiés ou utilisés séparément.

Registres des ports P0, P1, P2, et P3

Les registres P0, P1, P2 et P3 correspondent aux tampons des 4 ports de 8 bits du microcontrôleur. Pour sortir une donnée sur l'un de ces ports, il suffit d'écrire cette donnée dans le registre correspondant.

Registre de données série SBUF

Le tampon de communication série est composé de deux registres séparés, un pour l'émission, un autre pour la réception. Ces deux registres sont à la même adresse (SBUF = 99H). Quand une donnée est **"écrite"** dans SBUF, elle est placée dans le registre d'émission. Par contre une lecture de SBUF permet de récupérer une donnée située dans le registre de réception.

Registres des timers

Les paires de registres TH0 TL0, TH1 TL1, et TH2 TL2 sont des compteurs 16 bits utilisables pour le comptage ou la temporisation d'événements. L'utilisation de ces compteurs fait l'objet d'un chapitre particulier.

Registres de capture : RCAP2H et RCAP2L

La paire de registres RCAP2H et RCAP2L constitue un registre 16 bits pour le TIMER 2, de la version 8052, programmé en mode capture.

Dans ce mode, une transition d'état sur la broche T2EX provoque la copie du contenu des registres TH2 et TL2 dans les registres RCAP2H et RCAP2L. Le temporisateur 2 peut aussi fonctionner en temporisateur à **rechargement** automatique. Dans ce mode, RCAP2H et RCAP2L contiennent alors la valeur initiale de temporisation.

Registres de contrôle

Ces registres à fonction spéciale contiennent les bits de contrôle ou d'état pour le système d'interruption (IP et IE), pour les compteurs/temporisateurs (TMOD, TCON, T2CON), et pour le port série (SCON).

Modes de consommation réduite

Le fait de placer le microcontrôleur dans un mode de consommation réduite permet de conserver le contenu de la mémoire vive interne tout en limitant au strict minimum la consommation d'énergie.

Cette possibilité ne concerne que les versions réalisées en technologie CMOS. Les termes "**power-down**" et "**Idle**" désignent les deux modes de consommation réduite. Le terme de "mise en sommeil" est utilisé pour traduire l'idée contenue dans le mode de fonctionnement nommé "*Power-down*". Le terme "*Idle*" peut alors se traduire par "*mise en veille*".

La mise en sommeil et la mise en veille sont commandées par des bits du registre à fonction spéciale PCON.

PCON : Adresse directe 87 H

SMOD	—	—	—	GF1	GF0	PD	IDL

SMOD : Mis à 1, ce bit assure un doublement de la fréquence de communication du port série obtenue à partir du TIMER 1.
GF1 : Indicateur général sans fonction particulière.(1)
GF0 : Indicateur général sans fonction particulière.(1)
PD : Validation du mode Power-down.(1)
IDL : Validation du mode IDLE.(1)

(1) Ces bits n'existent que dans les versions réalisées en technologie CMOS.

Mise en sommeil (Power-down)

La mise en sommeil bloque l'oscillateur du microcontrôleur et donc, toutes les fonctions de celui-ci sont stoppées. Seuls les contenus de la RAM interne et des registres à fonction spéciale sont conservés. Les signaux ALE et PSEN sont figés à l'état bas.

Pour entrer dans ce mode, il suffit de mettre à 1, le bit PD du registre PCON par l'instruction suivante :

```
ORL  PCON,#00000010B  ;mise à 1 DE PD
```

Cette instruction sera la dernière à être exécutée avant que le microcontrôleur n'entre dans la mise en sommeil.

Lorsque le microcontrôleur est maintenu dans ce mode, il est possible de réduire la tension VCC (entre 2 volts et 5.5 volts), mais alors quelques précautions sont à prendre. La tension ne doit pas baisser avant que le mode de mise en sommeil ne soit établi, et doit être rétablie à sa valeur normale avant le "réveil" du microcontrôleur.

Sortie du mode

Le seul moyen de sortir de ce mode consiste à provoquer une initialisation externe par la broche RESET (broche RESET mis à 1). Cette initialisation modifie le contenu des registres à fonction spéciale mais laisse inchangé le reste de la RAM interne.

Mise en veille (idle mode)

La mise en veille consiste à isoler l'unité centrale de l'horloge, mais l'oscillateur continue de fonctionner. Tous les périphériques restent opérationnels. La réduction de consommation dépend d'ailleurs du nombre de périphériques qui ont été validés par le programmeur avant la mise en veille. Les broches correspondant aux ports conservent l'état qu'elles avaient au moment de l'entrée dans ce mode et les signaux ALE et PSEN sont forcés à 1. Les ports peuvent être utilisés comme entrées, les opérations de comptage d'événements extérieurs, de rechargement automatique, de capture et de comparaison restent possibles. Les interruptions à partir de sources externes sont prises en compte.

Ce mode permet donc de geler le processeur durant un certain temps dans l'attente par exemple d'un évènement extérieur. Cet évènement pourra provoquer la remise en service normal du microcontrôleur.

La mise en veille s'obtient par la mise à 1 du bit IDL par une instruction du genre :

```
ORL   PCON,#00000001B   ;mise à 1 de IDL
```

Cette instruction est la dernière à être exécutée avant la mise en veille. Il est possible de sortir de ce mode par demande d'interruption. Le programme de service de l'interruption sera exécuté normalement, puis après l'instruction RETI (qui clôture normalement ce programme de service) c'est la première instruction qui suit l'instruction de mise en veille qui sera exécutée. Dès la reconnaissance d'une interruption autorisée, le processeur reprend donc son fonctionnement normal.

Dans le cas où les deux bits PD et IDL sont mis à 1 simultanément, seule la fonction mise en sommeil est alors retenue.

9

Conception et utilisation des ports d'E/S

Les quatre ports P0,P1,P2 et P3 du 8051 constituent un ensemble de 32 lignes bidirectionnelles pouvant être utilisées aussi bien en entrées qu'en sorties indépendantes les unes des autres et ce, sans procédure d'initialisation particulière. Chacune de ces lignes bidirectionnelles est matérialisée par une broche du circuit intégré. Comme il en est fait mention au chapitre 3, la limitation du nombre de broches du circuit intégré a conduit le constructeur à affecter à certaines lignes de ces ports une fonction auxiliaire. La diversité des fonctions auxiliaires a conduit à une conception technologique différente pour chacun des quatre ports

Caractéristiques communes aux quatre ports

Le registre associé à chaque port est constitué de huit bascules "type D". Lors de l'initialisation du microcontrôleur, la sortie de ces huit bascules est positionnée à 1. Cette mise à 1 des broches d'entrée ou sortie est à prendre en compte lors de la conception d'un système ! En effet, si la ligne est utilisée comme sortie, cette sortie sera à l'état haut au moins tout le temps de la phase "RESET" du microcontrôleur et elle ne passera à l'état souhaité, par l'utilisateur, qu'à l'exécution d'une instruction d'écriture dans le registre correspondant au port dont fait partie cette sortie.

Par contre, lorsque la ligne doit être utilisée comme une entrée, cet état initial est indispensable à la bonne réception d'information extérieure. La structure générale du tampon de sortie relié à la broche du circuit intégré est telle que si ce tampon est porté à l'état bas, il "court-circuitera" toute variation d'état provenant d'un signal externe. Avant d'utiliser une ligne de port comme entrée, il est donc impératif de s'assurer que le bit lui correspondant dans le registre de port, soit bien à l'état 1, tel qu'il a été positionné par l'initialisation du contrôleur.

Cette remarque n'est pas inutile et peut permettre de comprendre, par exemple, un mauvais fonctionnement de l'interface série en mode réception : le bit 0 du registre P3 (P3.0) doit être à 1 pour que cette réception soit possible.

Structure générale d'un port d'entrée/sortie

Le couplage entre l'unité centrale proprement dite et les broches du circuit intégré est assuré par les trois sous-ensembles suivants :

— un registre à fonction spéciale portant le même nom que le port ;

— un étage amplificateur de sortie dont la structure diffère selon la fonction auxiliaire associée à chacun d'eux ;

— un tampon d'adaptation de niveau d'entrée.

Si la structure des étages amplificateurs de sortie diffèrent pour chacun des quatre ports, les figures 9.2, 9.3, 9.4, et 9.5 font apparaître une similitude de structure en ce qui concerne les sous-ensembles registre et tampon de lecture. L'organisation commune est résumée par la figure 9.1. La bascule type D mémorise la donnée en provenance du bus interne sous le contrôle du signal d'écriture issu de l'unité centrale. La sortie Q de cette bascule peut être mise en liaison avec le bus interne par une commande de lecture. Il est alors possible de lire le contenu du registre. Mais une autre liaison est possible entre la broche du circuit intégré et le bus interne de données. Cette liaison est commandée par un signal de lecture de la broche. Selon le type d'instruction utilisé l'unité centrale va donc pouvoir lire soit l'état du registre, soit l'état logique présent sur la broche du circuit.

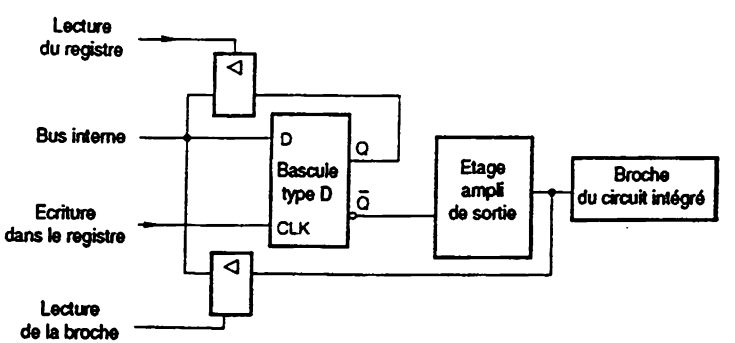


Figure 9.1 : Structure générale d'une des 8 lignes d'entrée/sortie de l'un des ports

Structure des ports P1, P2 et P3

La figure 9.2 donne une conception générale d'une ligne d'un de ces trois ports. Elle fait apparaître par rapport à la figure 9.1 le détail de l'étage de sortie. Pour ces trois ports, l'étage de sortie est équipé d'un transistor n1 et de trois autres transistors dont le montage équivalent est représenté par une résistance de rappel au potentiel $+V_{CC}$. La représentation donnée par cette figure ne fait pas apparaître une logique intermédiaire placée entre la sortie du registre et l'étage de sortie. Cette logique peut varier de constitution selon la fonction auxiliaire attribuée à chacun de ces trois ports.

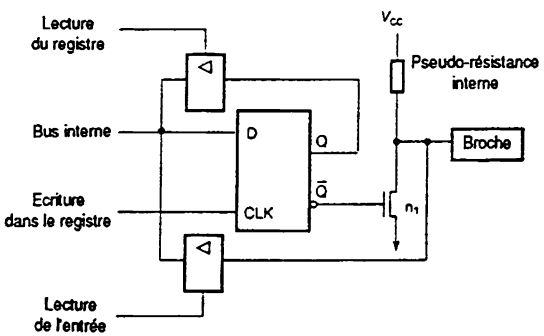


Figure 9.2 : Structure générale d'un étage de sortie des ports P1-P2 et P3

Chaque ligne peut être utilisée indépendamment des autres comme une entrée ou comme une sortie du type logique. En cas d'utilisation comme entrée, le bit du registre lui correspondant doit être alors dans un état logique 1. Par rapport à la figure 9.2 ceci se traduit par une sortie Q à 1 alors que son complément (Q barre) est à 0. Le transistor n1 est bloqué. La broche d'entrée est portée à un potentiel proche de $+ V_{CC}$ à travers la pseudo résistance de rappel. On note donc que si l'élément externe relié à cette entrée présente un état logique 0, il y aura circulation d'un courant sortant de cette entrée. Pour un port véritablement "bidirectionnel", une entrée devrait être dans un état de haute impédance. C'est pour cette raison que les ports P1, P2 et P3 sont considérés seulement comme "quasi bidirectionnels".

La réalisation de la résistance de rappel diffère selon la technologie utilisée.

Étage de sortie en technologie HMOS

La figure 9.3 représente la constitution détaillée d'un étage de sortie en technologie HMOS. Le transistor n3 constitue la voie basse et les transistors n1 et n2 constituent la voie haute ou pseudo résistance de rappel au $+V_{CC}$.

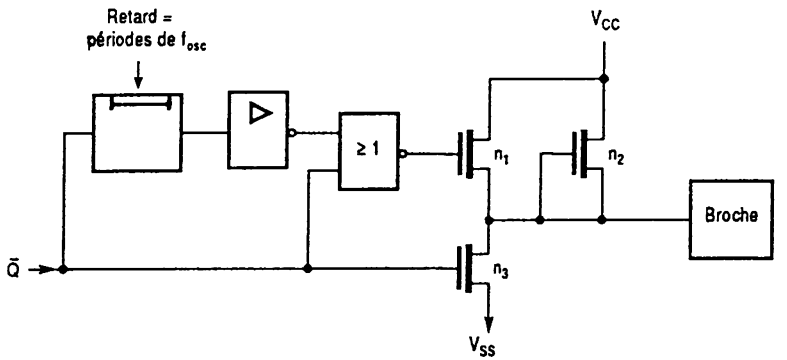


Figure 9.3 : Détail de l'étage de sortie en technologie HMOS

Le transistor n1, capable de fournir un fort courant, est activé seulement pendant deux périodes d'oscillateur. Cette activation a lieu lors d'une transition 0 vers 1. Le constructeur INTEL indique que la sortie peut fournir un courant supplémentaire de 30 mA durant la conduction de n1. Le transistor n2 délivre un courant plus faible représenté par le symbole I_{IL} dans les caractéristiques électriques du port. Le transistor n3, capable d'absorber un fort courant, est passant lorsqu'un état logique 0 de sortie est programmé dans le registre. Sa capacité à drainer un courant est spécifiée par la valeur I_{OL}. Il faut noter que ce transistor peut être détruit par une liaison directe entre la sortie et le potentiel $+V_{CC}$.

Étage de sortie en technologie CMOS

En technologie CMOS la résistance de rappel (figure 9.4) est réalisée par le couplage des trois transistors p1, p2 et p3. Ces trois transistors à effet de champ sont du type "canal P".

Lors d'une transition 0 vers 1 de la sortie, le transistor p1 est activé durant les deux périodes d'oscillateur S1P1 et S1P2. Ce transistor fourni alors un fort courant à la sortie. La valeur de ce courant est similaire au courant que peut absorber le transistor n1 pour un état de sortie 0. La conduction de p1 améliore la vitesse du changement d'état de la sortie.

Le transistor p2 reste passant tant que le bit du registre fixe un état logique 1 pour la broche de sortie. Ce transistor peut fournir un courant moins important que p1.

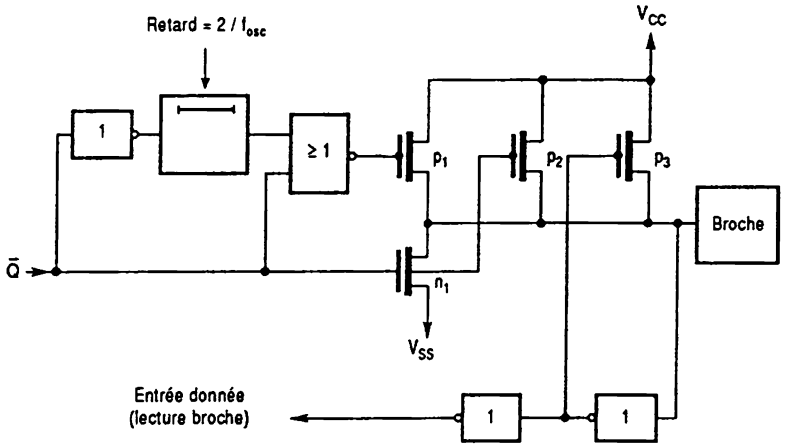


Figure 9.4 : Détail de l'étage de sortie en technologie CMOS

Le transistor p3 est du type "canal P" comme p1 et p2. Il est passant si le potentiel de la broche de sortie est supérieur à une tension comprise entre 1 et 1.5 V. Dans le cas où la ligne est utilisée comme entrée, ce transistor restera bloqué si la broche est forcée à un état logique 0 par un circuit externe. L'étage de sortie débitera vers l'entrée un courant limité à I_{IL} (courant que peut fournir le transistor p2). Par contre lorsque la ligne est utilisée comme sortie portée à un état logique 1, cette sortie pourra débiter un courant I_{TL} représentant la somme des courants débités par p2 et p3. Le transistor p3 permet donc de limiter la puissance dissipée lorsque la ligne est utilisée comme entrée et d'augmenter la valeur du courant délivré lorsque la ligne est utilisée comme sortie.

Structure du port P0

La structure interne du port P0 est détaillée par la figure 9.5 La fonction auxiliaire attribuée à ce port nécessite une logique de commutation et une conception particulière du tampon de sortie. Comparé aux trois autres ports, le port P0 est réellement bidirectionnel : il peut présenter un état logique "flottant" (haute impédance) lorsqu'il est considéré comme une entrée. Cette caractéristique est due à l'absence de résistance interne de rappel au potentiel $+V_{CC}$. En fait, le rôle de résistance de rappel est joué par le transistor à effet de champ "voie haute", mais celui-ci n'entre en action que pour des accès à une mémoire externe (utilisation en fonction auxiliaire du port). Pour garantir un état de haute impédance à l'entrée, il faut aussi

bloquer le transistor "voie basse". Ceci est obtenu par la présence d'un état logique 1 en sortie du registre. Lorsque P0 est utilisé comme simple port de sortie (figure 9.6), il faut le considérer comme du type à drain ouvert et donc prévoir la mise en place de résistances externes de rappel au $+V_{CC}$.

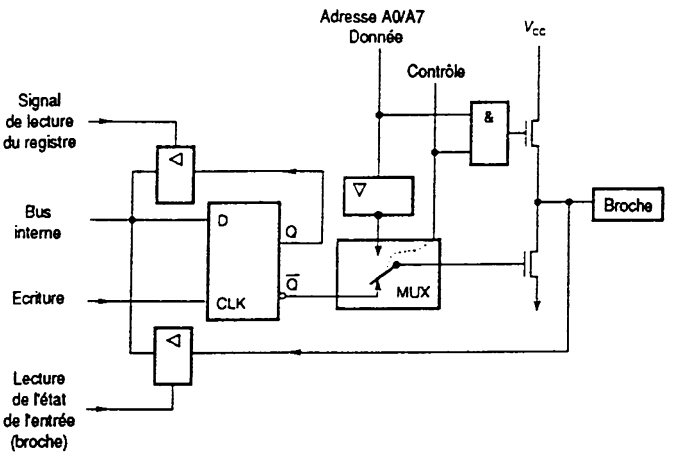


Figure 9.5 : Structure particulière du port P0

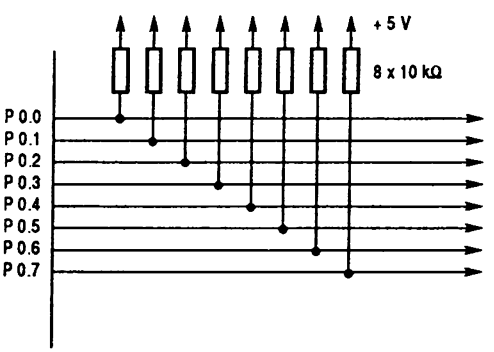


Figure 9.6 : Port 0 utilisé comme ensemble de 8 sorties

Lorsque P0 est utilisé comme bus de liaison à une mémoire externe (exécution d'instructions du genre MOVX par exemple), les sorties du registre sont automatiquement forcées à l'état 1.

Ports P0 et P2 comme bus de données et d'adresses

Comme on peut le constater sur les figures 9.5 et 9.7, l'étage de sortie peut se voir appliquer deux sources d'informations :

— les sorties du registre P0 ou le bus d'adresses lui même multiplexé au bus de données, pour le port P0 ;
— les sorties du registre P2 ou la partie haute (A8/A15) du bus d'adresses, pour le port P2.

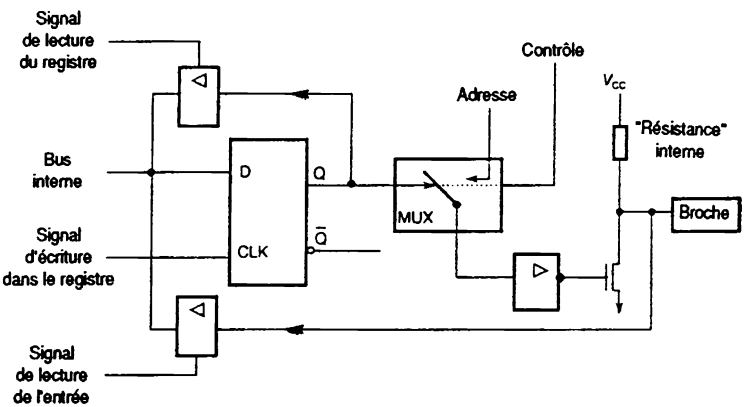


Figure 9.7 : Structure du port P2

La commutation est déterminée d'une façon interne. Elle est obtenue par un signal de contrôle issu d'une part, de l'état de la broche EA du microcontrôleur, et d'autre part, du type de l'instruction en cours d'exécution.

Lorsque ces ports sont configurés en bus d'adresses et de données, les registres à fonction spéciale P0 et P2 sont déconnectés des étages de sortie. Les 8 bits du registre P0 sont forcés à 1 alors que le contenu du registre P2 reste inchangé.

Fonctions auxiliaires du port P3

Selon la fonction auxiliaire attribuée à chaque ligne, on peut envisager une utilisation auxiliaire aussi bien en entrée qu'en sortie (figure 9.8). Pour assurer un fonctionnement correct de la ligne en fonction auxiliaire il est nécessaire que la sortie Q du registre soit maintenue à 1. Dans le cas contraire, le transistor de voie basse reste passant, bloquant ainsi la broche du circuit intégré à un potentiel proche de 0 V.

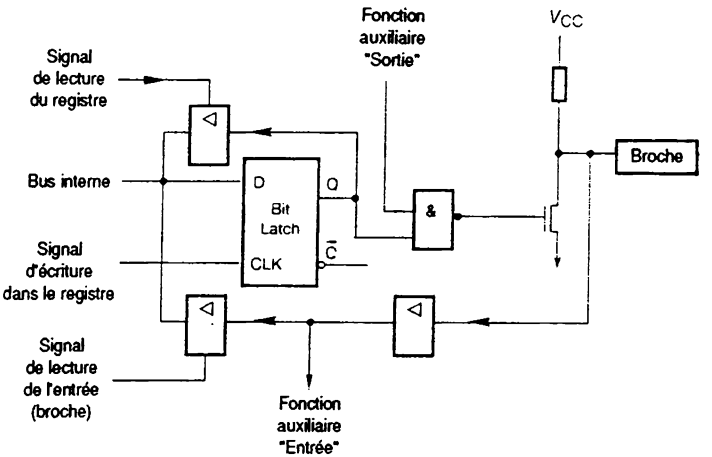


Figure 9.8 : Structure du port P3

P3.0	RXD	PORT SERIE réception de données en mode asynchrone; entrée et sortie de données en mode synchrone.
P3.1	TXD	PORT SERIE émission de données en mode asynchrone; sortie horloge en mode synchrone.
P3.2	$\overline{INT0}$	interruption externe 0, entrée de contrôle Timer 0.
P3.3	$\overline{INT1}$	interruption externe 1, entrée de contrôle Timer 1.
P3.4	T0	entrée de comptage pour Timer 0.
P3.5	T1	entrée de comptage pour Timer 1.
P3.6	$\overline{WR}$	signal d'écriture pour mémoire externe.
P3.7	$\overline{RD}$	signal de lecture pour mémoire externe.

Tableau 9.9 : Fonctions auxiliaires du port P3

Particularité logicielle concernant les quatre ports

La fin du paragraphe 9.2 précise le fait que certaines instructions vont permettre de lire l'état logique présent sur la broche du circuit intégré, et que d'autres instructions vont permettre de "lire" le contenu du registre associé au port. Ces dernières instructions sont nommées en anglais "read-modify-write instructions". La liste de ces instructions est donnée dans le tableau 9.10. Le résultat de l'exécution de ces instructions modifie le contenu du registre du port ou au moins un des bits de ce registre. Les autres instructions de lecture prennent en compte l'état présent sur l'entrée. Par exemple, l'instruction ANL P3,#01 (instruction figurant dans le tableau 9.10) récupère la valeur contenue dans le registre P3, effectue l'opération logique entre cette valeur et la

constante 01, puis recopie le résultat dans le registre P3. Dans le cas où l'on veut prendre en compte les états présents sur les entrées du port, il faut d'abord lire ces états puis effectuer le ET logique entre cette lecture et la constante :

MOV A, P3 ; état des entrées dans accumulateur

ANL A, #01 ; ET logique entre P3 et 01

Lorsque le port est utilisé comme sortie, on remarque donc que le contenu du registre de ce port constitue une mémoire des états "sorties" sur les broches du circuit intégré. Il n'est pas très sûr d'envisager de lire l'état d'une sortie directement sur la broche du circuit intégré, car le niveau électrique présenté par celle-ci dépend essentiellement du type de charge qui lui est appliqué. Une sortie de port, dans un état logique 1, reliée à une base de transistor, peut présenter une tension voisine de 0.7V par rapport à la masse. La tension lue sur la broche risque, dans ce cas, d'être interprétée comme un état logique 0 ! C'est dans le but d'éviter ces erreurs d'interprétation que les instructions du tableau 9.10 prennent comme source d'information le contenu du registre au lieu de prendre en compte l'état des broches.

Instruction	Fonction
ANL	ET LOGIQUE ex: ANL P0, A
ORL	OU LOGIQUE ex: ORL P2, #01110000B
XRL	OU EXCLUSIF ex: XRL P1, A
JBC	Saut si bit du port à 1 et mise à 0 du bit
CPL	Complémenter UN BIT du port
INC	Incrémenter de 1 le registre du port
DEC	Décrémenter de 1 le registre du port
DJNZ	Décrémenter de 1 et saut si résultat <> 0
MOV P.x,C	Copie du bit retenue C vers bit de port
CLR P.x	Mise à zéro d'un bit de port
SETB P.x	Mise à 1 d'un bit de port

Tableau 9.10 : Ces instructions "lisent" et "modifient" le port au niveau du registre de celui-ci

Chronologie des opérations sur un port

Ecriture sur un port (sortie)

Lorsqu'une opération provoque la modification du contenu du registre attribué au port, la nouvelle valeur est présente sur les sorties des bascules type D, durant la période S6P2 du dernier cycle machine nécessité par l'exécution de l'instruction.

Cependant, l'étage de sortie n'échantillonne la sortie du registre que sur les phases P1 de chaque période. En conséquence la nouvelle donnée ne parvient pas sur la sortie (broche du circuit intégré) avant la période S1P1 du cycle machine suivant (figure 9.11.a).

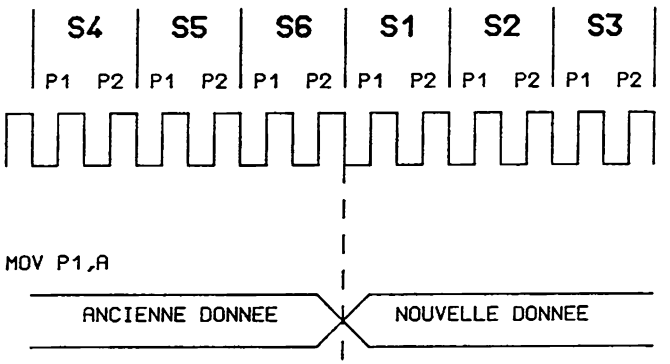


Figure 9.11a : Ecriture sur un port

Lecture d'un port (entrée)

Lorsqu'une instruction provoque la lecture des broches d'entrée d'un port, les états présents sur ces entrées sont échantillonnés durant la période S5. D'une façon plus précise, l'échantillonnage a lieu sur la phase S5P1 ou sur la phase S5P2 selon que le port est utilisé en fonction "PORT" ou dans sa fonction auxiliaire (figure 9.11.b). Il faut noter que toute variation d'entrée se produisant à un rythme supérieur à un cycle machine ne pourra pas être prise en compte.

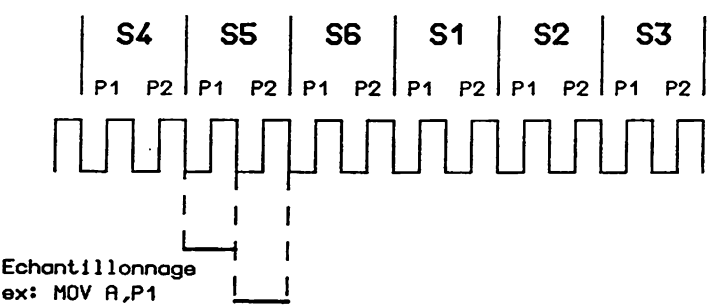


Figure 9.11b : Lecture d'un port

10

Description des timers

Généralités

Chaque timer est doté d'un compteur 16 bits accessible sous la forme de deux registres de 8 bits faisant partie des registres à fonction spéciale :

— TH0 et TL0 pour le timer 0,

— TH1 et TL1 pour le timer 1.

Ces deux timers sont configurables à partir des deux registres TMOD et TCON.

Un timer peut assurer deux fonctions distinctes :

— Le COMPTAGE, le compteur est alors incrémenté par la détection d'événements extérieurs. Ces événements sont des changements d'états appliqués aux broches dédiées à cette fonction (T0 = P3.4 ou T1 = P3.5).

— La TEMPORISATION, le compteur est incrémenté à partir du signal d'horloge du microcontrôleur soit directement soit à travers un circuit de division de la fréquence.

En plus du choix entre ces deux fonctions, les timers 0 et 1 connaissent quatre modes de fonctionnement différents, permettant de modifier le format du registre de comptage ou d'autoriser un rechargement automatique à partir d'une valeur de consigne. Le timer 1 peut faire office de générateur de fréquences de communication de l'interface de communication série.

La configuration d'un timer (choix de sa fonction et de son mode) est effectuée par programmation du registre de contrôle TMOD.

TMOD : Registre de contrôle du mode

Le tableau 10.1 précise le rôle de chaque bit du registre de contrôle du mode de fonctionnement des timers 0 et 1. Quatre bits sont dédiés à chaque timer.

TMOD : adresse directe 89 H

GATE	C/T	M1	M0	GATE	C/T	M1	M0
TIMER 1				TIMER 0			

GATE : GATE = 1, le TIMER x est validé seulement lorsque sa broche d'entrée "INTx" est à l'état 1 et que le bit "TRx" du registre TCON est à 1.
GATE = 0, le TIMER est validé si "TRx" = 1.

C/T : Permet la sélection de la fonction du TIMER :
C/T = 0 le TIMER est en fonction TEMPORISATEUR
C/T = 1 le TIMER est en fonction COMPTEUR à partir des événements présents sur la broche d'entrée "Tx".

M1	M0	Mode
0	0	compteur 13 bits compatible au microcontrôleur MCS-48
0	1	compteur 16 bits
1	0	compteur 8 bits à rechargement automatique. Le contenu de THx est rechargé dans TLx lorsque celui-ci repasse à 0.
1	1	TIMER 0 : Le registre TL0 devient un compteur 8 bits contrôlé normalement par les bits de contrôle du TIMER 0. Par contre, TH0 est configuré en temporisateur 8 bits contrôlé par les bits de contrôle du TIMER 1.
1	1	TIMER 1 : compteur 1 à l'arrêt.

Tableau 10.1 : Détail du registre de contrôle TMOD

Le bit GATE permet de valider la prise en compte des changements d'état ou des transitions détectées sur la broche externe INT0 ou INT1 (selon le timer utilisé). Le bit C/T permet de choisir entre la fonction "*compteur*" et la fonction "*temporisateur*". Les deux bits M0 et M1 assurent le choix parmi l'un des quatre modes de fonctionnement possibles.

C/T : Choix de la fonction du timer

Selon la valeur des bits C/T du registre TMOD, la fonction de chaque timer peut être modifiée.

Changer la fonction d'un timer consiste à changer la source des événements qui incrémentent le compteur du timer.

Fonction temporisation

Dans cette fonction, le compteur est incrémenté automatiquement à chaque cycle machine. Un indicateur est positionné lorsque le compteur, par phénomène de débordement, (*overflow*) repasse à la valeur 0. Ces indicateurs apparaissent dans le registre TCON (figure 10.6, page 120) sous les noms TF0 et TF1.

Fonction compteur

Le registre du timer est incrémenté sur une transition négative prise en compte à partir de l'entrée du timer (T0, T1, ou T2). Dans ce cas la broche d'entrée est échantillonnée durant la phase S5P2 de chaque cycle machine.

Lorsque l'échantillonneur "voit" un état haut puis, lors du cycle suivant, un état bas, le compteur est incrémenté. La nouvelle valeur apparaît dans le registre durant la phase S3P1 du cycle suivant la détection de la transition. Il faut donc deux cycles machines pour détecter et enregistrer un événement. Cette constatation limite la fréquence de comptage à une valeur vingt quatre fois moins importante que celle de l'oscillateur du microcontrôleur.

Description du mode opératoire (M1 et M0)

Le choix du mode est obtenu par position des bits M0 et M1 du registre TMOD.

Mode 0

Ce mode assure un fonctionnement des timers compatible à celui des timers des microcontrôleurs de la famille MCS-48. L'organisation fonctionnelle de ce mode est donnée figure 10.2. Le timer fonctionne alors en compteur 8 bits précédé par un prédiviseur de 32. Le registre du compteur est donc configuré en registre 13 bits : 5 bits sur la partie poids faible pour le prédiviseur et 8 bits sur poids fort pour le comptage. Lorsque le compteur repasse à 0, par débordement, l'indicateur d'interruption TFx est mis à 1. L'entrée de comptage est validée lorsque TRx (bit de contrôle du registre TCON) est à 1 et que GATE (bit de mode dans TMOD) est à 0 ou, lorsque l'entrée INTx présente un état 1.

Mode 1

Le mode 1 est identique au mode 0 excepté le fait que le registre du compteur utilise la pleine capacité des 16 bits sans prédiviseur. L'organisation fonctionnelle de ce mode est représentée par la figure 10.3.

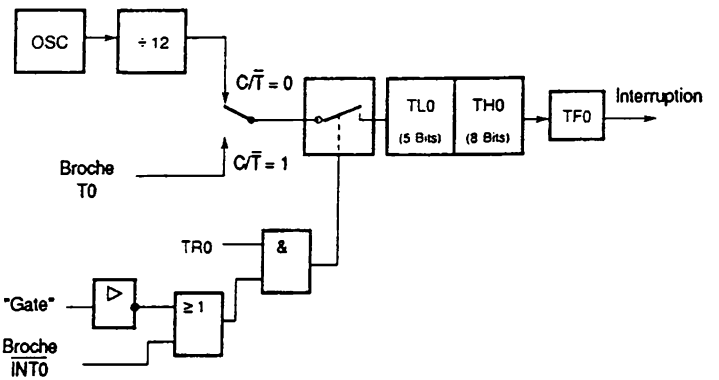


Figure 10.2 : Timer 0 ou 1 en mode 0 ***(d'après document Siemens)***

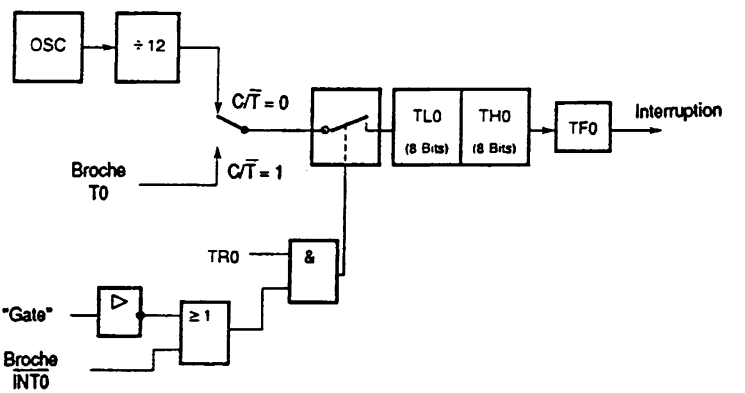


Figure 10.3 : Timer 0 ou 1 en mode 1 ***(d'après document Siemens)***

Mode 2

Dans ce mode le registre du timer est configuré en un compteur 8 bits à rechargement automatique (figure 10.4). La partie de poids faible TLx (TL0 ou TL1 selon le timer utilisé) sert de compteur. Lorsque ce compteur repasse à 0 par débordement, l'indicateur TFx est mis à 1 et le contenu de THx est rechargé dans le compteur TLx. Le contenu de THx, déterminé par programmation reste inchangé.

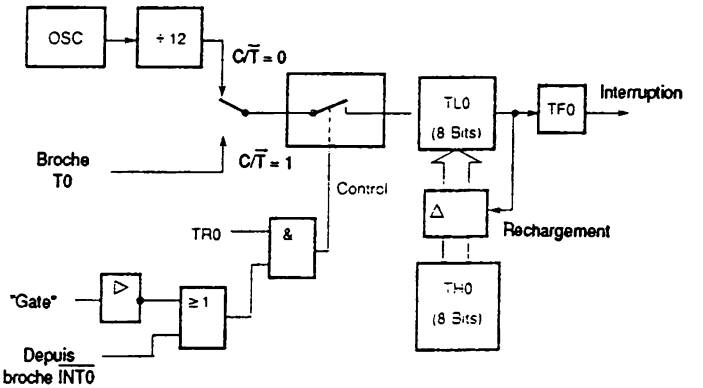


Figure 10.4 : Timer 0 ou 1 en mode 2

Mode 3

Le timer 1 en mode 3 est bloqué comme si son bit de contrôle TR1 était à 0.

Par contre pour le timer 0 (figure 10.5), les registres TL0 et TH0 sont configurés comme 2 compteurs de 8 bits. Le compteur TL0 utilise la logique de contrôle du timer 0 (C/T, GATE, TR0, INTO, et TF0) alors que le compteur TH0 est bloqué en fonction temporisateur sous le contrôle de TR1. Le compteur TH0 positionne l'indicateur TF1, et récupère donc la fonction d'interruption du timer 1.

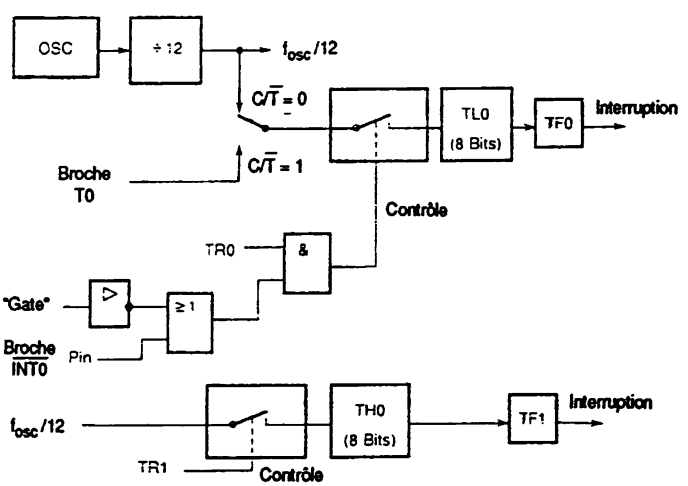


Figure 10.5 : Timer 0 en mode 3 : 2 compteurs 8 bits

Le mode 3 est destiné aux applications nécessitant un compteur supplémentaire de 8 bits. Avec le timer 0 configuré en mode 3, le 8051 possède alors un timer 16 bits (le timer 1), et deux timer de 8 bits comme l'indique la figure 10.5. Cette remarque n'est valable que pour des applications où le timer 1 n'utilisera pas sa ressource d'interruption puisque son indicateur (TF1) est alors câblé sur la sortie du compteur 8 bits réalisé à partir de TH0.

Registre de contrôle TCON

Le registre TCON peut être adressé au niveau du bit. Les quatre bits de poids fort concernent directement les timers 0 et 1. Les quatre bits de poids faible concernent les deux sources d'interruption externe INT0 et INT1.

TCON : Adresse directe 88 H adressable au niveau bit

8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 : Indicateur de débordement du TIMER 1. Mis à 1 automatiquement lorsque le compteur repasse à 0. Si l'interruption correspondante est autorisée, cet indicateur est remis à 0 automatiquement lorsque le sous-programme d'interruption est exécuté.
TR1 : Bit de déclenchement du TIMER 1. Il doit être positionné par logiciel à 1 (pour lancer) ou 0 (pour arrêter) le TIMER.
TF0 : Indicateur de débordement du TIMER 0. Identique à TF1.
TR0 : Bit de déclenchement du TIMER 0. même fonctionnement que TR1.
IE1 : Indicateur de transition sur la source externe INT1. Cet indicateur est mis à 1 automatiquement lorsqu'une transition est détectée sur la borne INT1. Lors du service de l'interruption cet indicateur est remis à 0.
IT1 : Sélection du type d'événement devant déclencher l'interruption depuis la source INT1.
IT1 = 0 interruption sur détection d'un niveau 0 sur INT1.
IT1 = interruption sur transition négative (1 vers 0).
IE0 : Indicateur de transition sur la source externe INT0.
Fonctionnement identique à IE1.
IT0 : IT0 = 0 interruption sur détection d'un niveau 0 sur INT0.
IT0 = 1 interruption sur détection d'une transition négative.

Tableau 10.6 : Registre de contrôle TCON

Timer 2 du 8052

Le timer 2 n'existe que dans les versions 8052.

C'est un compteur 16 bits qui peut fonctionner soit en compteur d'événements, soit en temporisateur sous le contrôle d'un registre à fonction spéciale T2CON d'adresse directe C8H. Trois modes de fonctionnement sont possibles. Il s'agit des modes

"capture", "rechargement", et générateur de fréquence pour communication série. Ce dernier mode de fonctionnement utilise une définition sur 16 bits. L'utilisation du timer 2 en remplacement du timer 1 permet donc un choix plus grand de vitesse de communication sur le port série.

Registre T2CON

T2CON adresse directe C8H adressable au niveau bit

CFH	CEH	CDH	CCH	CBH	CAH	C9H	C8H
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C / $\overline{T2}$	CP/$\overline{RL2}$

TF2: *Dépassement de capacité. Mis à 1 lorsque le Timer 2 passe de la valeur FFFFH à la valeur 0.*

EXF2: *Indicateur externe. Mis à 1 lorsqu'une capture ou un rechargement est provoqué par transition 0 vers 1 sur broche T2EX et que EXEN2 = 1. Si l'interruption du Timer 2 est autorisée et que EXF2=1 alors le sous-programme placé à l'adresse 2BH est exécuté. La remise à 0 de EXF2 doit être assurée par programmation.*

RCLK: *Bit de contrôle de l'horloge de réception. RCLK =1 positionné à 1, le port série, programmé en mode 1 ou 3, utilise l'impulsion de dépassement du Timer 2 comme signal d'horloge de réception. RCLK =0, c'est l'impulsion de dépassement du Timer 1 qui joue ce rôle.*

TCLK: *Contrôle de l'horloge d'émission. TCLK = 1, le port série, (en mode 1 ou 3), utilise l'impulsion de dépassement du Timer 2 comme signal d'horloge d'émission. TCLK = 0, c'est l'impulsion de dépassement du Timer 1 qui joue ce rôle.*

EXEN2: *Validation du mode externe. EXEN2 = 1 : autorisation de capture ou de rechargement provoqué par une transition négative détectée sur broche T2EX. Ceci n'est possible que si le Timer 2 n'est pas utilisé comme horloge du port série (TCLK et RCLK à 0). Le fait de forcer EXEN2 à 0 permet d'ignorer les événements présents sur la broche T2EX.*

TR2 : *Contrôle du fonctionnement. La mise à 1 de TR2 provoque le démarrage du Timer.*

C/$\overline{T2}$: *Sélection du mode de fonctionnement du Timer 2 :*

C / $\overline{T2}$ =0 fonction "temporisateur" ;

C / $\overline{T2}$ =1 fonction "compteur d'événements externes".

En mode temporisateur, c'est l'horloge (du microcontrôleur) divisée par 12 qui commande l'incrémentation du Timer. En mode compteur, seules des transitions négatives peuvent être détectées comme événements externes.

CP/$\overline{RL2}$: *Sélection pour capture ou rechargement. CP/$\overline{RL2}$ = 1 le mode capture est sélectionné. La conjugaison EXEN2=1 et détection d'une transition négative sur T2EX permet une capture : la valeur contenue par le registre 16 bits du Timer 2 est placée dans les registres RCAP2H et RCAP2L. Ce mode de capture n'est possible que si RCLK et TCLK sont à 0. Dans le cas contraire, c'est le mode rechargement automatique qui est sélectionné.*

Fonctionnement en mode capture

Illustré par la figure 10.8, ce mode permet de "capturer" ou mémoriser la valeur courante du timer 2. Le timer 2, est un compteur de 16 bits composé des registres TH2 et TL2. L'opération de capture recopie la valeur de TH2 dans le registre RCAP2H et la valeur de TL2 dans RCAP2L.

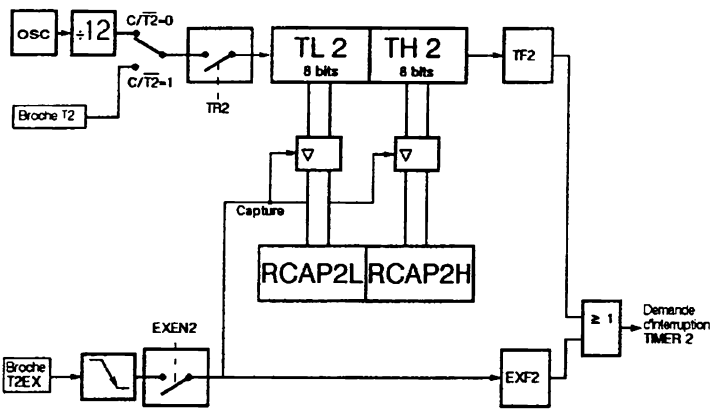


Figure 10.8 : Timer 2 en mode capture

Dans le mode "capture" l'état du bit EXEN2 détermine deux options de fonctionnement.

EXEN2 = 0 : le timer 2 peut alors fonctionner en compteur ou en temporisateur 16 bits. Lors du dépassement de capacité du registre 16 bits, le bit TF2 est automatiquement positionné à 1, ce qui peut produire une interruption (à condition qu'elle soit autorisée par le programmeur).

EXEN2 = 1 : le fonctionnement général reste identique à celui qui vient d'être décrit. La différence réside dans la prise en compte d'événements extérieurs. Lors d'une transition logique 1 vers 0 (transition négative) sur la broche T2EX les contenus des registres TH2 et TL2 sont copiés respectivement dans les registres RCAP2H et RCAP2L. Dans le même temps, l'indicateur EXF2 est positionné à 1 et peut ainsi permettre le déclenchement d'une interruption. Il faut remarquer que les indicateurs TF2 et EXF2 sont affectés du même vecteur d'interruption (adresse 002BH). Lors du traitement d'interruption, le programmeur devra déterminer par test, l'événement ayant provoqué l'interruption.

Fonctionnement en "rechargement automatique"

Comme pour le mode "capture", ce mode de fonctionnement est contrôlé par le bit EXEN2.

EXEN2 = 0 : le compteur évolue jusqu'au dépassement de capacité, et positionne l'indicateur TF2, le compteur, c'est-à-dire TH2 et TL2, est alors rechargé de la valeur contenue par RCAP2H et par RCAP2L. Ces deux derniers registres doivent bien sûr, avoir été initialisés d'une donnée par le programmeur. Ce mode de fonctionnement peut être utilisé comme générateur de fréquence pour la communication série soit seulement pour la réception (RCLK = 1), soit seulement pour l'émission (TCLK = 1), ou encore pour les deux (RCLK = 1 et TCLK = 1).
EXEN2 = 1 : le fonctionnement en rechargement automatique reste similaire. En plus, une transition négative sur la broche T2EX provoque un rechargement anticipé et provoque la mise à 1 de EXF2.

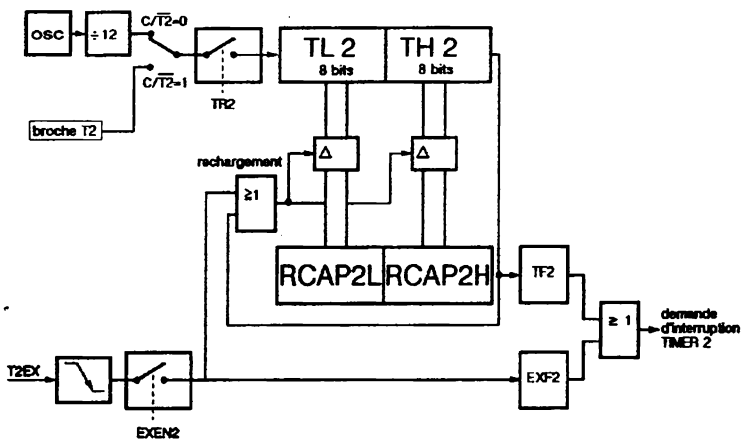


Figure 10.9 : Timer 2 en mode rechargement automatique

Lorsque le mode "*rechargement automatique*" est utilisé comme générateur de fréquence pour la communication série (RCLK+TCLK = 1), les sous-ensembles du timer 2 prennent l'organisation précisée par la figure 10.10. Il faut remarquer que le compteur est alors commandé par une fréquence plus élevée ($F_{OSC}/2$ au lieu de $F_{OSC}/12$). La fréquence de communication FC est donnée par la formule :

$$FC = \frac{F_{OSC}}{32.\ [65536 - (RCAP2H,\ RCAP2L)]}$$

où RCAP2H et RCAP2L représentent la valeur 16 bits de rechargement du compteur.

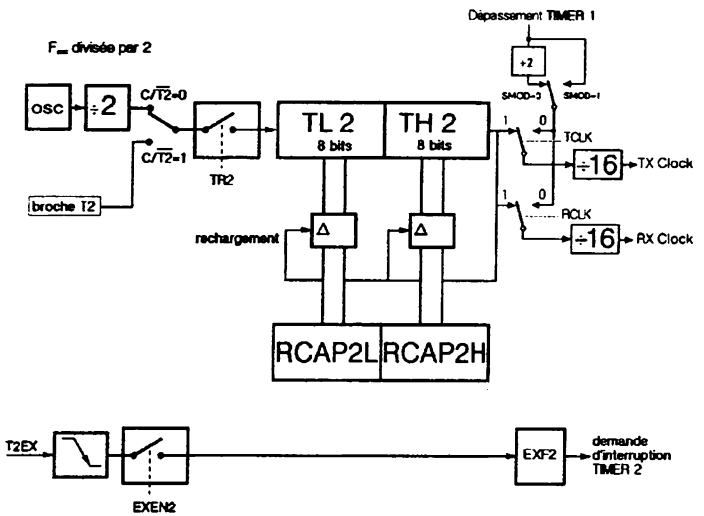


Figure 10.10 : Timer 2 en mode rechargement automatique si RCLK + TCLKG = 1

Il faut noter que dans cette utilisation, l'indicateur TF2 n'est plus opérationnel. Seule une détection de transition d'état émanant de la broche T2EX peut alors être source d'interruption.

Exemple de programmation du timer 2

A titre d'exemple, on se propose de décrire le module d'initialisation du timer 2 tel que, celui-ci provoque une interruption toutes les millisecondes. Le programme d'interruption, non décrit, sera placé à partir de l'adresse 002BH.

Le mode "rechargement automatique" est retenu pour cette application. Il faut donc définir les valeurs à placer dans les registres RCAP2H et RCAP2L pour obtenir une temporisation d'une milliseconde. Le compteur est incrémenté tous les cycles machines, soit toutes les microsecondes si le quartz de l'oscillateur est de 12 MHz. Ce compteur provoque l'interruption lorsque son contenu dépasse la valeur 65 535. Il faut donc partir de la valeur :

$$65\,536 - 1\,000 = 64\,536$$

Cette valeur de 16 bits est à répartir entre le registre de poids fort RCAP2H et le registre de poids faible RCAP2L suivant la formule suivante :

(RCAP2H x 256)+(RCAP2L) = 64 536

contenu de RCAP2H : 64 536/256 = 252

contenu de RCAP2L : 64536 — (256 x 252) = 24

Pour assurer une première période d'une milliseconde, il est souhaitable d'initialiser TH2 et TL2 de ces mêmes valeurs.

Une fois les registres initialisés, il faut déclencher le temporisateur en mettant à 1 le bit TR2 et autoriser la source d'interruption du timer 2.

INITIALISATION DU TIMER 2
MODE RECHARGEMENT AUTOMATIQUE
65 536-(RCAP2H x 256+RCAP2L)= 1 ms

```
MOV   RCAP2H,#252        ;252 x 256
MOV   RCAP2L,#24         ;+24
MOV   TH2,#252
MOV   TL2,#24
MOV   T2CON,#00000100B   ;LANCER TIMER TR2 = 1
MOV   IE,#10100000B      ;AUTORISER IT TIMER 2
RET
```

Générateur de fréquence de communication série

Utilisation du timer 1

Le timer 1 peut être utilisé comme horloge pour la communication série. Il doit être configuré alors en mode 2, c'est-à-dire en compteur huit bits à rechargement automatique. Dès que le registre TL1 repasse à 0, il est rechargé de la valeur contenue dans le registre TH1. La vitesse de communication FC est définie par la formule suivante :

$$FC = \frac{K \times f_{OSC}}{32 \times 12(256 - TH1)}$$

où f_{OSC} = fréquence de l'oscillateur du microcontrôleur
et K = coefficient de valeur 1 ou 2.

La valeur de K est définie par l'état du bit SMOD du registre PCON :

si SMOD = 0, alors $K = 1$,
si SMOD = 1, alors $K = 2$.

D'une façon pratique, il est nécessaire de définir la valeur à placer dans le registre TH1 en fonction d'une fréquence FC. La formule devient :

$$TH1 = 256 - \frac{K \times f_{OSC}}{384 \times FC}$$

Exemple de calcul : On se propose de définir la valeur de TH1 pour obtenir une vitesse de communication normalisée de 19 200 Bauds avec une tolérance de 2 %. La fréquence de l'oscillateur est de 12 MHz et K est choisi comme égal à 2. Il faut placer dans TH1 la valeur théorique de :

$$256 - \frac{2 \times 12 \times 10^6}{384 \times 19\,200} = 252{,}744\,79$$

La valeur à placer dans TH1 ne peut être qu'un entier huit bits. En optant pour une valeur de 252 par défaut, le calcul de vérification de FC conduit à une fréquence de 15 625 Bauds. Par contre le calcul de FC pour la valeur par excès de TH1 = 253 donne une fréquence de 20 833 Bauds. Ces deux valeurs ne sont pas comprises dans la tolérance fixée.

Cet exemple fait donc apparaître l'impossibilité d'obtenir, à partir du timer 1, une vitesse de communication de 19 200 Bauds avec un quartz de 12 MHz comme base de temps pour l'oscillateur.

Le constructeur INTEL propose l'utilisation d'un quartz de 11,059 MHz pour résoudre ce problème. Un nouveau calcul avec cette valeur de quartz donne :

$$256 - \frac{2 \times 11\,059 \times 10^6}{384 \times 19\,200} = 253{,}00005$$

En tenant compte de la tolérance de 2 %, ce calcul reste valable pour tout quartz dont la fréquence d'oscillation serait comprise entre 10,9 MHz et 11,3 MHz. Un quartz de 11 MHz peut donc parfaitement convenir.

Dans l'exemple de calcul, il a été choisi un coefficient K = 2, ce qui correspond à un état logique 1 du bit SMOD. Ce bit étant initialisé à l'état 0 par le microcontrôleur, il est donc nécessaire de positionner ce bit dans le registre PCON qui n'est accessible qu'en adressage direct. L'instruction

ORL PCON,#80H

positionne à 1 le bit de poids fort (SMOD) de PCON situé à l'adresse 87H

Le programme complet d'initialisation du timer 1, fonctionnant comme générateur de vitesse de communication, est le suivant :

Configuration du timer 1

— en mode 2 : rechargement automatique de TL1 par TH1

— TH1 = 253

— lancé lorsque TR1 = 1 (TR1 bit 6 de TCON)

Définition de TMOD :

TIMER 0				TIMER 1			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
0	0	1	0	0	0	0	0

soit la valeur 20H (le timer 0 n'étant pas pris en compte).

```
MOV     TMOD,#20H     ; timer 1 en MODE 2
MOV     TH1,#253      ; fréquence de 19200 bauds
ORL     PCON,#80H     ; avec SMOD = 1 pour K = 2
SETB    TR1           et lancer le timer
```

Le tableau 10.11 donne les valeurs à affecter à TH1 pour obtenir certaines vitesses de communication standard utilisée par exemple par la norme RS 232 ou la norme MIDI (communication entre instruments musicaux).

Fréquence de communication	Fréquence de l'oscillateur	Valeur de SMOD	C/T	TIMER 1	
				Mode	TH1
19200	11.059 MHz	1	0	2	FDH
9600	11.059 MHz	0	0	2	FDH
4800	11.059 MHz	0	0	2	FAH
2400	11.059 MHz	0	0	2	F4H
1200	11.059 MHz	0	0	2	E8H
300	6 MHz	0	0	2	CCH
110	6 MHz	0	0	2	72H
MIDI					
31250	12 MHz	0	0	2	FFH

Tableau 10.11 : Fréquence de communication à partir du TIMER 1

Utilisation du timer 2 du 8052

Lorsqu'on utilise le timer 1 en mode "rechargement automatique", le réglage de la fréquence est limité à une résolution de 8 bits. Cette limitation a été mise en évidence dans l'exemple précédent : une horloge de 12 MHz ne permettait pas d'obtenir une fréquence de communication de 19 200 bauds à 2 % près.

Le timer 2, lorsqu'il est utilisé en mode "rechargement automatique" (voir page 123), fonctionne sur 16 bits. Le timer 1 peut être remplacé partiellement ou totalement par le timer 2 dans la fonction d'horloge du port série. Ce remplacement est programmé à partir des bits RCLK et TCLK du registre T2CON :

RCLK	TCLK	Fonction
0	0	le timer 1 contrôle le port série.
0	1	le timer 2 contrôle la vitesse d'émission et le timer 1 contrôle la vitesse de réception.
1	0	le timer 2 contrôle la vitesse de réception et le timer 1 contrôle l'émission.
1	1	le timer 2 contrôle émission et réception.

Pour le timer 2, la fréquence de communication est donnée par la formule :

$$FC = \frac{f_{OSC}}{32 \times [65\,536 - (\text{RCAP2H, RCAP2L})]}$$

où RCAP2H,RCAP2L représente un entier de 16 bits.

Ces deux registres servent au rechargement de TH2 et de TL2.

De cette formule on peut définir une formule pratique qui permettra de définir les valeurs à placer dans RCAP2H et RCAP2L pour une fréquence de communication choisie :

$$(\text{RCAP2H, RCAP2L}) = 65\,536 - \frac{f_{OSC}}{32 \times FC}$$

A partir de cette formule, déterminons les contenus de RCAP2H et RCAP2L pour obtenir une fréquence de communication FC de 19 200 bauds si F_{OSC} est égale à 12 MHz :

$$65\,536 - \frac{12 \times 10^6}{32 \times 19\,200} = 65\,516{,}469$$

soit la valeur entière de 65516 ou FFECH. Il faut donc placer la valeur FFH dans RCAP2H et la valeur ECH dans RCAP2L. Avec la valeur arrondie de FFECH la valeur de FC est de 18750 Bauds. Cette valeur est très proche de la tolérance inférieure à 2 % (18816 Bauds). En toute rigueur, FC reste en dehors de la tolérance fixée, même si la valeur obtenue reste plus acceptable que la valeur obtenue avec le timer 1.

Le tableau 10.12 donne des valeurs standards de FC et les valeurs à placer dans RCAP2H et RCAP2L pour les obtenir :

Fréquence de communication	Fréquence de l'oscillateur	TIMER 2	
		RCAP2H	RCAP2L
9600	12 MHz	FF	D9
4800	12 MHz	FF	B2
2400	12 MHz	FF	64
1200	12 MHz	FE	C8
300	12 MHz	FB	1E
110	12 MHz	F2	AF
300	6 MHz	FD	8F
110	6 MHz	F9	57

Tableau 10.12 : Fréquence de communication à partir du TIMER 2

Le programme suivant est un exemple d'initialisation de communication par le port série. La vitesse de communication est la même en émission et en réception : 9600 Bauds. Le port série est initialisé en mode 1 (voir chapitre suivant) :

```
MOV     SCON, #01011010B  ; initialise port série
MOV     RCAP2H, #0FFH     ; valeur de rechargement pour TH2
MOV     RCAP2L, #0D9H     ; valeur pour TL2
MOV     T2CON, #34H       ; TCLK et RCLK à 1 et lancer timer
```

Un autre exemple où la réception est pilotée par le timer 1 à une vitesse de 4 800 bauds alors que la vitesse de l'émission de 2 400 bauds est obtenue grâce au timer 2 :

```
MOV     SCON, #01011010B  ; initialise port série
MOV     RCAP2H, #0FFH     ; valeur de rechargement pour TH2
MOV     RCAP2L, #064H     ; valeur pour TL2
MOV     T2CON, #00010100B ; TCLK à 1 et lancer timer 2
MOV     TMOD, #00100000B  ; 8 bits auto reload timer 1
MOV     TH1, #0FAH        ; valeur pour 4800 bauds
SETB    TR1               ; lancer timer 1
```

11

Interface de communication série

Présentation

L'interface de communication série est matériellement accessible par la fonction secondaire de deux lignes du port P3. Il s'agit de la borne RXD (fonction secondaire de P3.0) pour la réception, et de la borne TXD (deuxième fonction de P3.1).

Le port série fonctionne en *"full duplex"*, c'est-à-dire en réception et transmission simultanée. Il possède aussi un tampon de réception qui autorise la réception d'un octet avant même la lecture du précédent. Cependant la lecture du premier octet doit être effectuée avant la réception du dernier bit du deuxième octet. Les registres de réception et de transmission sont tous deux accessibles, à la même adresse de la RAM interne, désigné par le symbole SBUF dont la valeur est 99H. Une écriture à cette adresse provoque un accès au registre de transmission, alors qu'une lecture du contenu de cette adresse permet de récupérer la donnée du registre de réception.

La commande et le contrôle du port série s'effectuent grâce au registre à fonction spéciale SCON situé à l'adresse 98H. Le port série peut fonctionner selon quatre modes :

MODE 0 : Registre à décalage
Le port série est utilisé comme un registre à décalage au format de 8 bits. Les données série entrent ou sortent par la broche RXD La broche TXD est utilisée comme horloge dont la fréquence est obtenue par division par 12 de la fréquence de l'oscillateur du microcontrôleur.

MODE 1 : Le port série fonctionne en UART 8 bits avec un bit de départ et un bit de stop. 10 bits sont ainsi transmis (par la borne TXD) ou reçus (sur la borne RXD). En réception, le bit de STOP est placé dans le registre SCON. La vitesse de communication est variable. Elle est définie par programmation du TIMER 1 pour le 8051 et des TIMER 1 et 2 pour le 8052. Cette particularité autorise alors des vitesses différentes pour la réception et la transmission.

MODE 2 : Le port série est configuré en UART de format 9 bits. La donnée est encadrée d'un bit de start et d'un bit de stop. Pour la transmission, le neuvième bit provient du registre de contrôle SCON, en réception ce neuvième bit sera placé dans ce même registre. La vitesse d'émission/réception est programmable : soit 1/32 ou 1/64 de la fréquence oscillateur du microcontrôleur.

MODE 3 : Le mode 3 est identique au mode 2. La seule différence réside dans le fait que la vitesse d'émission/réception est variable : elle est générée par le compteur 1 (1 ou 2 sur le 8052).

Registre de contrôle

Le registre de fonction spéciale SCON assure le contrôle et indique l'état de l'interface série. SCON permet entre autre, la sélection du mode de communication.

SCON : Adresse directe 98 H adressable au niveau du bit

9F	9E	9D	9C	9B	9A	99	98
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

mode de fonctionnement du port série :

SM0	SM1	Mode	Description	fréquence de communication
0	0	0	registre à décalage	fosc./12
0	1	1	UART 8 bits	variable
1	0	2	UART 9 bits	fosc./64 ou fosc./32
1	1	3	UART 9 bits	variable

SM2 : Communication multiprocesseur possible en mode 2 et 3.
En mode 2 ou 3, si SM2=1, alors RI ne sera activé que si le 9ème bit est un 1.
En mode 1, si SM2 = 1, alors RI ne sera activé qu'à la réception d'un bit de stop valide.
En mode 0, SM2 doit être positionné à 0.

REN : Mis à 1 par programme REN autorise la réception série.

TB8 : correspond au 9ème bit à transmettre en mode 2 et 3.

RB8 : correspond au 9ème bit reçu en mode 2 et 3. En mode 1, si SM2 = 0, RB8 correspond au bit de stop reçu.

TI : est l'indicateur d'émission. Il est automatiquement mis à 1 :
à la transmission du 8ème bit en mode 0 ;
à la transmission du bit de stop dans les autres modes.
Avant une opération d'émission, il doit être remis à 0 par programme.

RI : Indicateur de réception. Il est automatiquement mis à 1 ; à la réception du 8ème bit en mode 0,
à la réception du bit de stop dans les autres modes (exception précisée dans la description de SM2).
RI doit être remis à 0 par programme.

Fonctionnement en registre à décalage (mode 0)

Les données série entrent et sortent par la broche RXD. La broche TXD représente alors l'horloge de décalage. Le format d'émission et de réception est de 8 bits (bit de poids faible en premier). La vitesse de communication est fixée à 1/12 de la fréquence de l'oscillateur du microcontrôleur. La figure 11.2 présente un diagramme fonctionnel simplifié du port série en mode 0.

La transmission est initialisée par l'instruction qui utilise l'adresse SBUF comme adresse destination (exemple MOV SBUF, A). Le signal "Écriture dans SBUF" qui a lieu sur la dernière phase du cycle d'exécution de l'instruction (S6P2), provoque le chargement d'un "1" dans la neuvième position du registre à décalage de transmission et lance le cycle de transmission. Un cycle machine (douze périodes d'horloge) après l'apparition de ce même signal, la commande interne SEND est activée. SEND valide la sortie du registre à décalage sur la broche P3.0 (RXD) et valide aussi l'horloge de décalage sur P3.1 (TXD).

L'horloge de décalage est basse pendant S3, S4 et S5 de chaque cycle machine et à l'état haut pendant S6, S1 et S2. A l'instant S6P2 de chaque cycle machine, lorsque SEND est actif, le contenu du registre de transmission est décalé d'une position vers la droite. Lorsque les bits de données sortent à droite, des 0 sont introduits à gauche. Quand le bit le plus significatif de l'octet de donnée est en position de sortie, alors le 1 qui a été initialement chargé en neuvième position est juste à gauche du bit le plus significatif et toutes les autres positions sont occupées par des 0. Cette condition indique au bloc de contrôle TX l'ordre d'un dernier décalage suivi de la désactivation de SEND et de la mise à 1 de T1. L'ensemble de ces opérations a duré dix cycles machine après le signal d'écriture dans SBUF.

La réception est initialisée par la condition REN=1 et RI=0. A la période S6P2 du prochain cycle machine, l'unité de contrôle RX inscrit les bits 11111110 dans le registre de réception et la phase d'horloge suivante active RECEIVE. RECEIVE active l'horloge de décalage sur la borne P3.1. L'horloge de décalage change d'état aux phases S3P1 et S6P1 de chaque cycle machine.

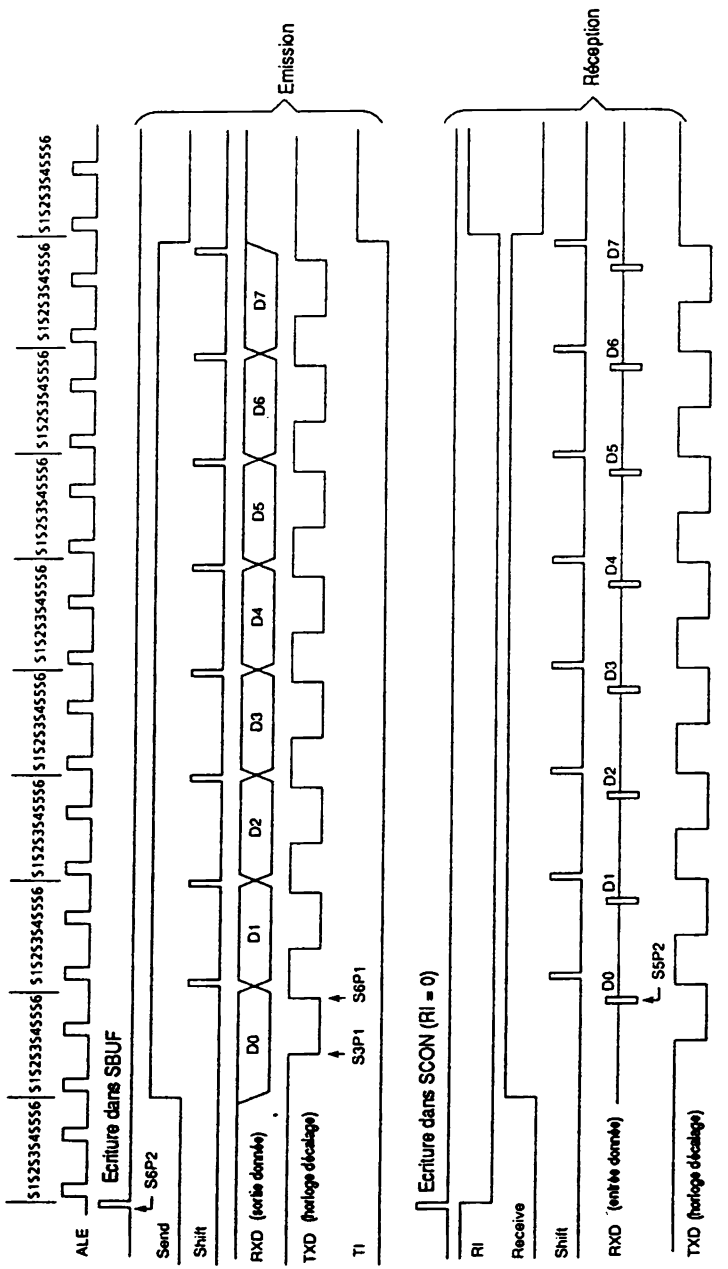


Figure 11.1 : Chronologie des opérations en mode 0 ***(d'après document SIEMENS)***

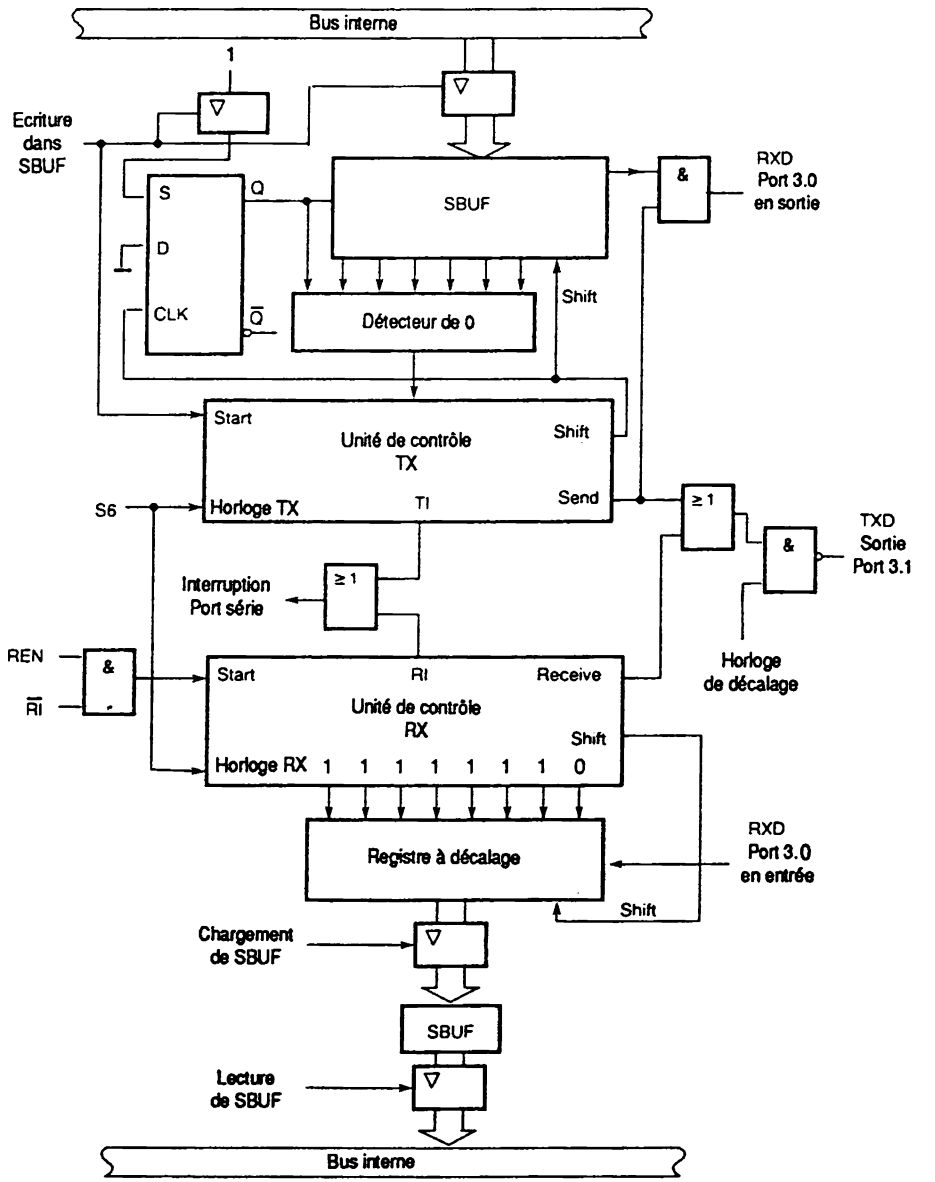


Figure 11.2 : Logigramme du port série en mode 0 *(d'après document SIEMENS)*

Sur la phase S6P2 de chaque cycle machine pendant que RECEIVE est activé, le contenu du registre à décalage est déplacé d'une position vers la gauche. La valeur qui vient depuis la gauche est échantillonnée sur la borne P3.0 pendant la phase S5P2 du même cycle machine.

Les bits de données "chassent" vers la gauche les "1" initialement chargés dans le registre de réception. Lorsque le 0 se retrouve en dernière position à gauche, cette situation indique au bloc de contrôle RX l'ordre d'un dernier décalage suivi du chargement de SBUF.

En considérant comme origine du temps l'instant où l'indicateur RI a été forcé à 0, c'est sur la phase S1P1 du dixième cycle machine que la bascule RECEIVE est remise à 0 et que l'indicateur RI est positionné à 1.

Fonctionnement en mode 1

L'interface série fonctionne alors en UART 8 bits à fréquence variable. 10 bits sont transmis (par la borne TXD) ou reçus (sur la borne RXD) : un bit de START ("0" logique), 8 bits de données (poids faible en premier) et un bit de STOP ("1" logique). En réception, le bit de STOP est placé dans RB8 du registre SCON.

Pour le 8051, la vitesse de transmission est déterminée par le TIMER 1. Pour le 8052 cette vitesse peut être déterminée comme pour le 8051 mais aussi à partir du compteur 2 ou encore grâce aux deux ce qui autorise alors des vitesses différentes pour la réception et la transmission. La figure 11.3. présente un diagramme fonctionnel simplifié du port série en mode 1.

La transmission est initialisée par l'instruction qui utilise SBUF comme registre de destination. Le signal d'écriture dans SBUF provoque le chargement d'un 1 dans la neuvième position du registre de transmission et indique à l'unité de contrôle TX qu'une transmission est demandée. La transmission commence sur la phase S1P1 du cycle machine suivant (l'envoi du bit est synchronisé sur le compteur diviseur par 16 et non pas sur le signal d'écriture dans SBUF). La transmission commence avec l'activation de SEND qui présente sur TXD le bit START . Un bit plus tard, DATA est activé, ce qui valide la sortie du registre TXD.

Comme les bits sortent par la droite, des "0" sont introduits par la gauche, quand le bit de données le plus significatif se retrouve en position de sortie dans le registre à décalage, alors le "1" qui avait été initialement chargé en neuvième position se retrouve en deuxième position suivi par une suite de "0". Cette condition indique à l'unité de contrôle TX l'ordre d'un dernier décalage, la désactivation de SEND et la mise à 1 de TI.

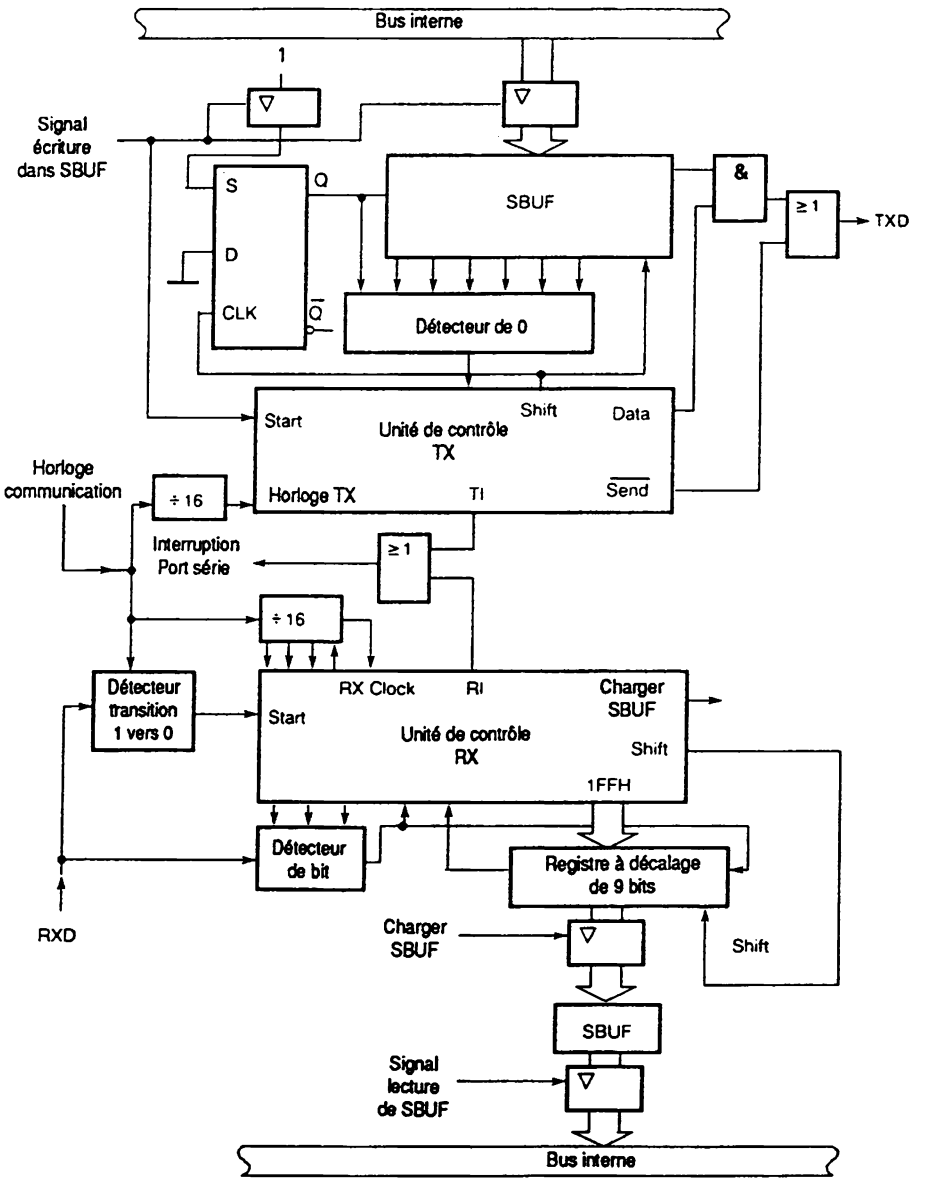


Figure 11.3 : Port série en mode 1 ***(d'après document SIEMENS)***

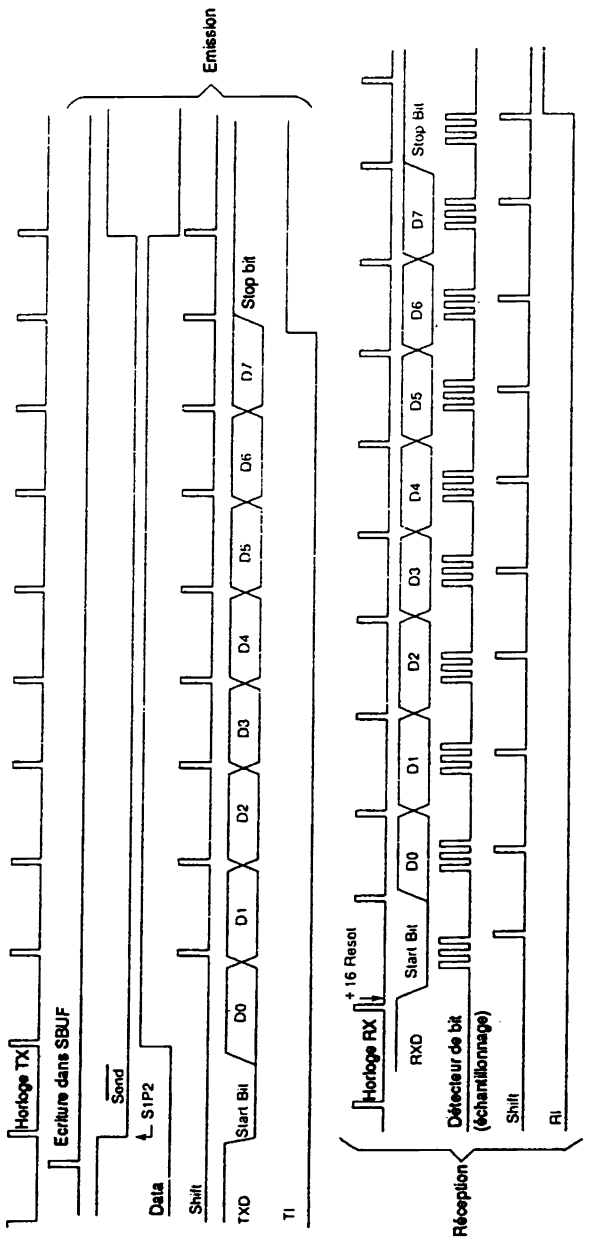


Figure 11.4 : Chronologie des opérations en mode 1 ***(d'après document SIEMENS)***

La réception est initialisée par la détection d'une transition de 1 vers 0 sur la broche RXD. Pour cette raison, RXD est échantillonné à une fréquence de 16 fois supérieure à la fréquence de réception. Dès la détection de cette transition le compteur par 16 est immédiatement initialisé et la valeur 1FFH est écrite dans le registre à décalage.

Les seize périodes du compteur par 16 permettent une division en seizièmes de la période de réception d'un bit. Aux septième, huitième et neuvième périodes de ce compteur le détecteur de bit échantillonne la valeur de RXD. La valeur retenue est le valeur "vue" au moins deux fois sur les trois. Cette méthode améliore la réjection du bruit. Si le premier bit détecté n'était pas un "0" (bit de START) le circuit de réception serait réinitialisé dans l'attente d'une nouvelle transition de 1 vers 0.

Dans le cas où le bit de START est valide, celui-ci est décalé vers l'entrée du registre à décalage et la réception des bits suivants est possible. Comme les bits arrivent depuis la droite, la sortie des 1 se fait par la gauche. Quand le bit de START se retrouve en position de sortie (dans le registre 9 bits pour le mode 1), l'unité de contrôle RX exécute un dernier décalage, charge la donnée dans SBUF et RB8 et positionne à 1 l'indicateur RI. Le signal de chargement de SBUF, deRB8 et le positionnement de RI n'ont lieu que si, et seulement si, la condition suivante est présente durant le dernier décalage :

RI = 0 et (SM2 = 0 ou STOP bit = 1)

Si cette condition n'est pas vraie la trame de réception est perdue et RI n'est pas positionné.

Si la condition est vraie, le bit de STOP est placé dans RB8 et les 8 bits de données sont chargés dans SBUF. A cet instant, que la condition ait été remplie ou pas, l'unité reprend l'attente d'une transition 1 vers 0 sur RXD.

Fonctionnement en mode 2 et 3

L'interface série fonctionne en UART dont le format des données est de 9 bits.

Onze bits sont transmis ou reçus (par les bornes TXD ou RXD). Un bit de départ (start bit = 0), 8 bits de données, un neuvième bit programmable, et un bit de stop (stop bit = 1). Pour la transmission, le neuvième bit (programmé dans TB8 du registre SCON) peut prendre les valeurs 0 ou 1, ou bien, par exemple servir de bit de parité si on lui affecte la valeur de l'indicateur P de PSW par l'intermédiaire de TB8. En réception ce neuvième bit sera placé dans RB8 de SCON.

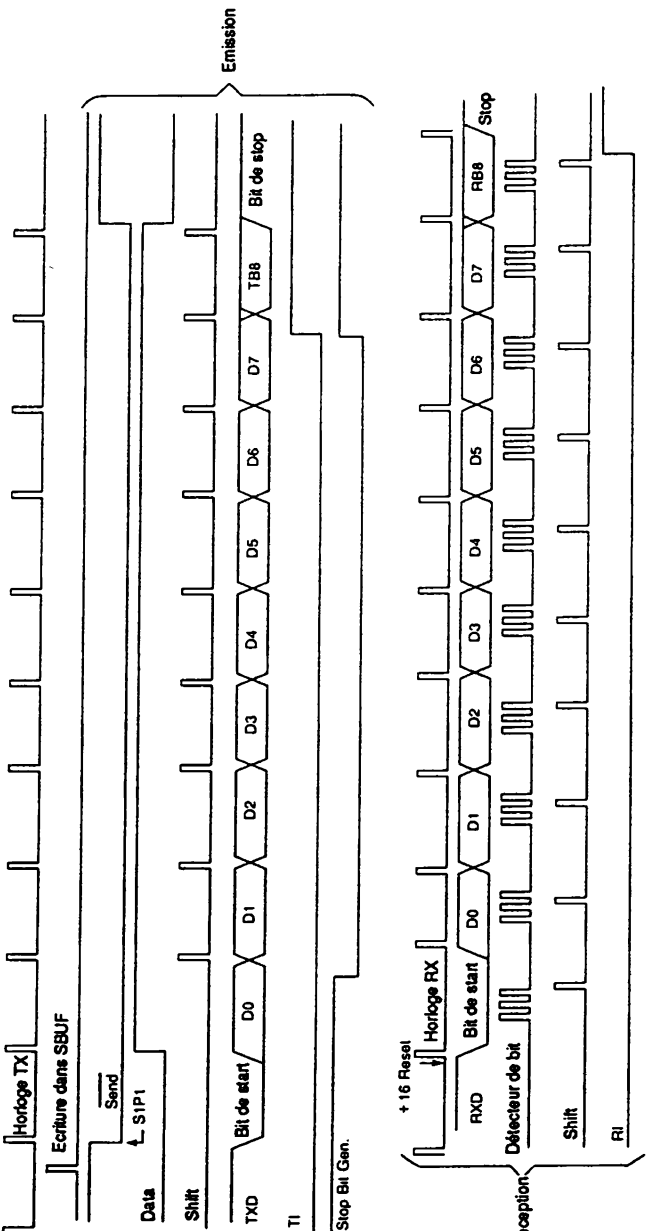


Figure 11.5 : Chronologie des opérations en mode 2 et 3 *(d'après document SIEMENS).*

Pour le mode 2, la vitesse d'émission et de réception est programmable : soit 1/32 ou 1/64 de la fréquence oscillateur du microcontrôleur. Le choix d'une des deux fréquences est déterminé par l'état de SMOD, bit 7 du registre PCON (Power Control Register). Ce bit est automatiquement initialisé à l'état 0 et permet d'obtenir une fréquence de communication 64 fois plus petite que celle de l'oscillateur du microcontrôleur. Lorsque l'utilisateur positionne ce bit à 1, la fréquence de communication est doublée.

Pour le mode 3, la vitesse d'émission et de réception est obtenue à partir du compteur/temporisateur 1 (1 ou 2 pour le 8052). Les figures 11.6. et 11.7. donnent les diagrammes fonctionnels du port série dans ces deux modes.

La transmission est initialisée par chaque instruction qui utilise SBUF comme registre de destination. Le signal d'écriture vers SBUF charge dans la neuvième position du registre à décalage, le neuvième bit de la donnée situé dans TB8 et indique à l'unité de contrôle TX qu'une transmission est demandée. La transmission commence à la période S1P1 du cycle machine suivant le passage à 0 du compteur diviseur par 16.

La transmission commence avec l'activation de SEND qui place le bit de START sur TXD. Un bit plus tard, DATA est activé, ce qui valide le bit de sortie du registre à décalage pour TXD. La première impulsion de décalage apparaît un bit plus tard.

Cette première impulsion introduit le 1 du bit STOP en neuvième position du registre à décalage. Puis c'est une suite de 0 qui est introduite depuis la gauche lors des décalages. Lorsque le bit de TB8 est en position de sortie, le bit de stop juste à sa gauche, tous les autres bits du registre à décalage sont à 0. Cette condition indique à l'unité de contrôle TX l'ordre d'un dernier décalage, de désactiver SEND et de positionner à 1 l'indicateur TI. Cette situation se produit à la douzième révolution du compteur diviseur par 16 après écriture dans SBUF.

La réception est initialisée par la détection d'une transition 1 vers 0 sur la broche RXD. Pour ce faire, la broche RXD est échantillonnée à une fréquence 16 fois supérieure à la fréquence de réception qui a été établie. Lorsque la transition a été détectée, le compteur diviseur par 16 est réinitialisé et la valeur 1FFH est écrite dans le registre à décalage d'entrée.

Les seize périodes du compteur par 16 permettent une division en seizièmes de la période de réception d'un bit. Aux septième, huitième et neuvième périodes de ce compteur le détecteur de bit échantillonne la valeur de RXD. La valeur retenue est le valeur "vue" au moins deux fois sur les trois. Cette méthode améliore la réjection du bruit. Si le premier bit détecté n'était pas un "0" (bit de START) le circuit de réception serait réinitialisé dans l'attente d'une nouvelle transition de 1 vers 0. Dans le cas où le bit de START est valide celui-ci est décalé vers l'entrée du registre à décalage et la réception des bits suivants est possible.

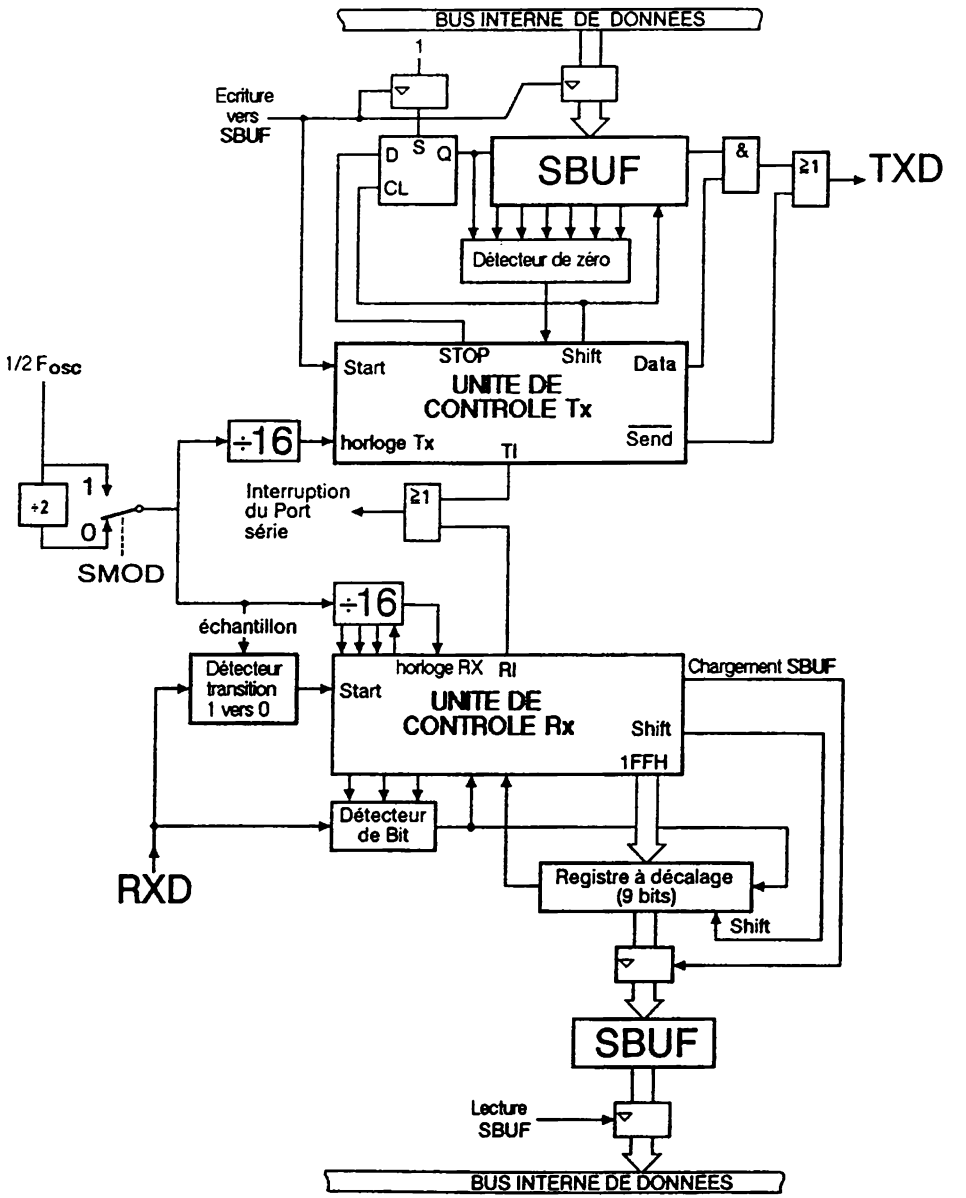


Figure 11.6 : Port série en mode 2 ***(d'après document SIEMENS).***

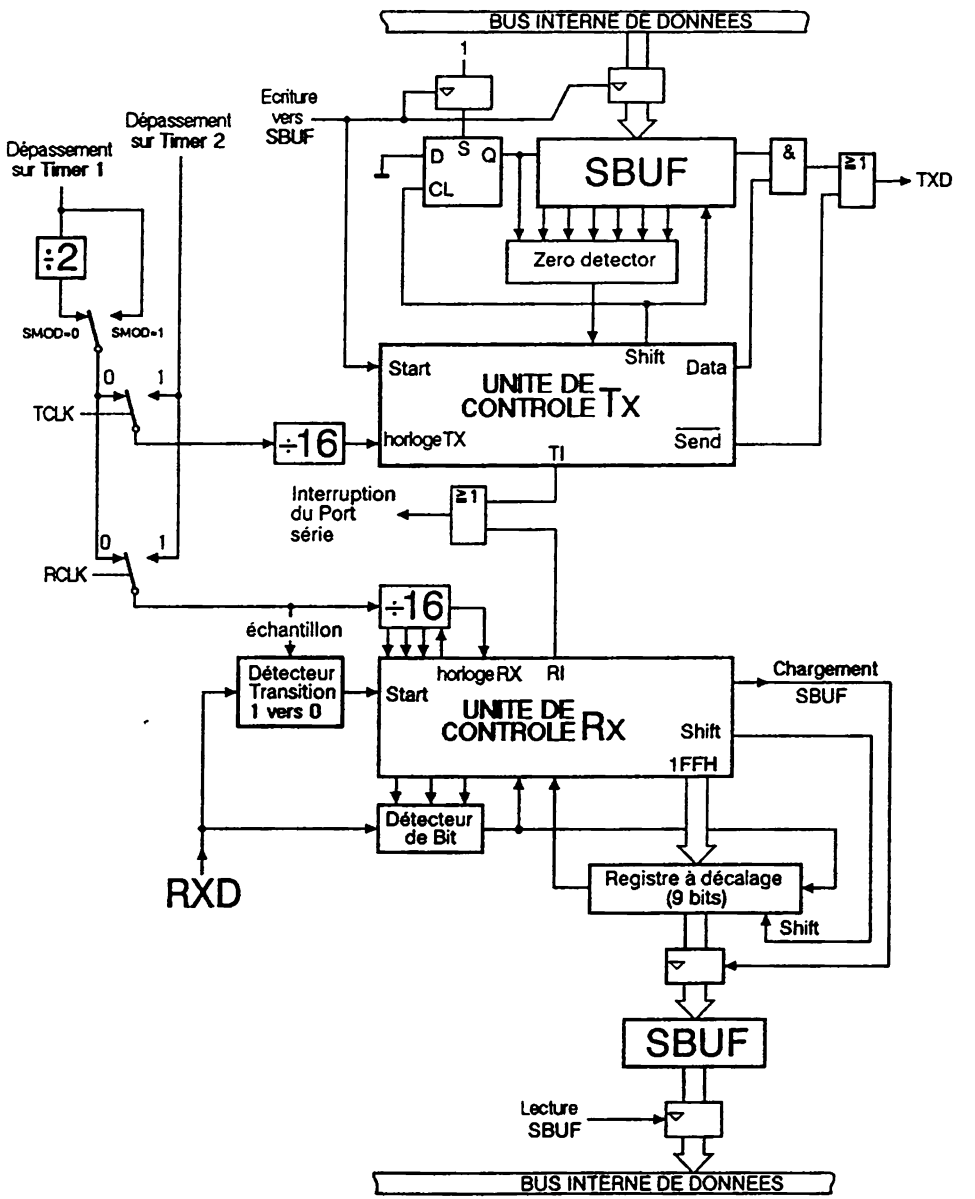


Figure 11.7 : Port série en mode 3 ***(d'après document SIEMENS)***

Comme les bits arrivent depuis la droite, la sortie des 1 se fait par la gauche. Quand le bit de START se retrouve dans la position la plus à gauche (dans le registre 9 bits pour les modes 2 et 3), l'unité de contrôle RX exécute un dernier décalage. Le chargement de la donnée dans SBUF et dans RB8 et la mise à 1 de l'indicateur RI n'ont lieu que si et seulement si la condition suivante est présente durant le dernier décalage :

RI = 0 et (SM2 = 0 ou neuvième bit = 1)

Si cette condition n'est pas vraie la trame de réception est perdue et RI n'est pas positionné.

Si la condition est vraie, le neuvième bit est placé dans RB8 et les huit premiers bits sont chargés dans SBUF. Un "bit plus tard", que la condition ait été remplie ou pas, l'unité reprend l'attente d'une transition 1 vers 0 sur RXD.

Communication multiprocesseur

Les modes 2 et 3 sont adaptés à une communication entre microcontrôleurs. Dans ces modes, 9 bits de données sont reçus (neuvième bit dans RB8), suivis d'un bit de stop. Le port série peut être programmé pour produire une interruption seulement si RB8 = 1. Cette configuration est possible en positionnant à 1 le bit SM2 de SCON.

Un exemple de protocole d'échange est donné ci-dessous

Quand le processeur maître veut transmettre un bloc de données à l'un des processeurs esclaves, il commence par envoyer un octet d'adresse dont le neuvième bit est à 1. Par opposition, tous les bits de données seront caractérisés par un neuvième bit à 0. Avec SM2 à 1, aucun esclave ne sera interrompu par un octet de données. Par contre, un octet "adresse" peut interrompre tous les esclaves. Ainsi chaque esclave analyse l'adresse envoyée. L'esclave adressé remet à 0 son bit SM2 et se prépare à recevoir les données qui vont suivre. Les autres laissent leur bit SM2 à 1 et retournent à leur tâche.

Exemples de programmes de gestion du port série

Initialisation du port série

Soit à réaliser une communication série au format 8 bits à une vitesse de 4 800 bauds pour un microcontrôleur 8031 dont la fréquence du quartz est de 11 059 MHz.

D'après les indications données dans le paragraphe 11.2 il faut choisir le mode 1. Dans ce mode, le bit SM2 peut être mis à 0 ou à 1. Si ce bit est mis à 1, l'indicateur de réception ne sera positionné à 1 que pour une réception se terminant par un bit de STOP valide. La réception est autorisée par la mise à 1 de REN. Les autres bits du registre SCON sont initialisés comme l'indique le tableau récapitulatif suivant :

SCON 98H :

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	1/0	1	1	0	1	0

Bien que non utilisé dans le mode 1, le bit TB8 peut être mis à 1. Le bit RB8 mis à 0, sera modifié lors d'une réception. TI est mis à 1 pour indiquer que le tampon d'émission est disponible.

La fréquence de communication de 4800 bauds sera obtenue à partir du timer 1. Par initialisation du registre TMOD on définit un mode de rechargement automatique sur le Timer 1. Le registre TH1 du compteur est chargé de la valeur FAH (voir tableau 10.11 de la page p. 127). Cette valeur sera rechargée automatiquement dans TL1 mais il n'est pas inutile d'initialiser TL1 afin que la fréquence d'horloge soit stable dès le lancement du Timer.

TMOD :

TIMER 1				TIMER 0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
0	0	1	0	X	X	X	X

GATE = 0 : GATE inactive

C/T = 0 le TIMER est en fonction **temporisateur**

M1 = 1 ET M0 = 0 : compteur 8 bits à rechargement automatique. Le contenu du registre TH1 est rechargé dans le registre TL1 lorsque la valeur de celui-ci repasse à 0 ; d'où le programme d'initialisation :

```
MOV   SCON,#01011010B     ;
MOV   TMOD,#00100000B     ; 8 BIT AUTO RELOAD TIMER 1
MOV   TH1,#0FAH           ; VALEUR POUR 4800 BAUDS
MOV   TL1,#0FAH           ;
SETB  TR1                 ; ACTIVE L'HORLOGE
```

Réception et émission

La réception d'un code valide est indiquée par la mise à 1 de l'indicateur RI. Une émission est possible lorsque le tampon de sortie est vide (TI=1)

Les sous-programmes suivants restent simples mais n'utilisent pas les avantages d'une gestion sous interruption.

```
EMISSION :   JNB  TI,EMISSION    ; ATTENDRE QUE TI=1 : TAMPON VIDE
             MOV  SBUF,A         ; PUIS ENVOI DU CODE DANS A
             CLR  TI             ; REMISE INDICATEUR A 0
             RET
RECEPTION :  JNB  RI,RECEPTION   ; ATTENDRE RECEPTION COMPLETE
             MOV  A,SBUF         ; PUIS PLACER CODE DANS ACCUMULATEUR
             CLR  RI             ; REMETTRE INDICATEUR A 0
             RET                 ET RETOUR
```

12

Les interruptions

Le noyau 8051 possède cinq sources d'interruption, alors que la version 8052 en comporte six comme l'indique la figure 12.1. Toutes ces sources d'interruption sont du type "masquable", c'est-à-dire autorisables ou non par logiciel. Après initialisation toutes les interruptions sont inhibées.

Le registre IE est dédié à l'autorisation ou l'inhibition individuelle des sources d'interruption.

Le mécanisme d'interruption est vectorisé sur adresse fixe. Ces adresses sont situées dans la partie basse de la mémoire programme.

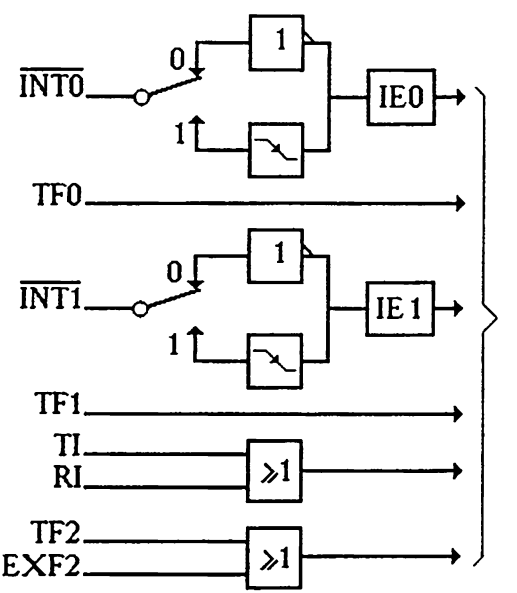


Figure 12.1 : Sources d'interruption (TF2 et EXF2 n'existent que dans les versions 8052)

Table des vecteurs

La logique d'implantation des vecteurs d'interruption est la suivante :

à l'adresse 0000 de la mémoire programme correspond la première instruction à exécuter après une initialisation (RESET = 1). Le constructeur a réservé 3 octets pour cette première instruction, valeur correspondant à une instruction de saut du type "**LJMP adr**".

C'est donc à l'adresse 0003 que se situe le premier emplacement réservé aux sous-programmes devant gérer les différentes sources d'interruption du programme principal. Chaque adresse correspondant à l'un de ces vecteurs est espacée de huit octets. Cet espace est dans la plupart des cas simplement suffisant à recevoir la "tête" du sous-programme de service de l'interruption. La table suivante précise les sources d'interruption, l'indicateur correspondant et l'adresse du vecteur propre à cette source :

Source d'interruption	Indicateur	Adresse du vecteur
INT0	IE0	0003H
TIMER 0	TF0	000BH
INT1	IE1	0013H
TIMER 1	TF1	001BH
PORT SERIE	RI+TI	0023H
TIMER 2	TF2+EXF2	002BH

Tableau 12.2 : Adresse des vecteurs d'interruption

Exemple d'implantation logicielle

L'exemple ci-dessous donne une idée générale de la mise en place de deux sous-programmes d'interruption correspondant aux sources **timer** 1 et **port serie** :

```
ORG     0           ; code à placer à l'adresse 0
LJMP    INISYSTEM   ; saut à INISYSTEM après un RESET
                    ; cette instruction occupe trois
                    ; octets en mémoire.
ORG     1BH         ; code suivant à 001BH
PUSH    PSW         ; 2 octets
PUSH    ACC         ; 2 octets
LJMP    IT_TIMER1   ; 3 octets suite du sous-programme
NOP                 ; 1 octet pour compléter
PUSH    PSW         ; ici l'adresse est 23H
PUSH    ACC         ; tête du sous-programme du port
LJMP    IT_SERIE    ; série puis saut vers suite...
NOP                 ; un NOP si int. TIMER utilisée
```

```
        .....
IT_TIMER1 :
        .....                   ; ici suite d'instructions du sous-
        .....                   ; programme pour gestion TIMER1
        .....                   ; cette suite se termine par...
        POP     ACC
        POP     PSW
        RETI
IT_SERIE :
        .....                   ; idem pour le port série
        .....
        POP     ACC
        POP     PSW
        RETI
INISYSTEM :
        .....                   ; initialisation et programme princ.
```

Exécution d'une interruption

Lorsque le processeur accepte une requête d'interruption, un saut automatique au sous-programme de service correspondant à la source est exécuté. Pour plus de précision le saut est comparable à un appel de sous-programme, c'est-à-dire qu'avant d'exécuter le branchement au sous-programme, le processeur sauvegarde automatiquement sur la pile l'adresse de la prochaine instruction qui devait normalement être exécutée. Un indicateur interne est positionné pour signaler le déroulement d'une interruption, puis le saut est exécuté. Un sous-programme de service d'interruption doit obligatoirement se terminer par l'instruction particulière de retour RETI. Elle permet tout comme l'instruction RET de récupérer depuis la pile l'adresse à laquelle le programme interrompu doit être repris, mais aussi de mettre à jour l'indicateur interne autorisant ainsi la prise en compte d'une autre interruption de même niveau.

Autorisation et priorité des interruptions

A l'initialisation du microcontrôleur, toutes les sources d'interruption sont inhibées par la remise à zéro des bits du registre IE. Afin d'utiliser le mécanisme des interruptions, il est donc nécessaire de commencer par autoriser la ou les sources d'interruption utilisées par l'application. La fonction des 8 bits du registre IE est donnée dans le tableau 12.3.

Ce registre fait partie de la RAM interne adressable au niveau du bit, il est donc possible par une seule instruction, de modifier un seul de ces bits sans modifier l'état des autres. Cette précision justifie le rôle du bit EA. En effet, quel que soit l'état des

7 autres bits, il est possible de suspendre la prise en compte de toute interruption en positionnant EA à 0, puis la remise à 1 de ce même bit autorise à nouveau la prise en compte des interruptions définies par l'état des 7 autres bits.

IE : Adresse directe A8H et adressable au niveau du bit

AF	AE	AD	AC	AB	AA	A9	A8
EA	—	ET2	ES	ET1	EX1	ET0	EX0

position	symbole	fonction
IE.7	EA	Inhibition de toutes les interruptions
IE.6	—	sans fonction pour 8051 et 8052
IE.5	ET2	interruption TIMER 2
IE.4	ES	interruption PORT SERIE
IE.3	ET1	interruption TIMER 1
IE.2	EX1	interruption EXTERNE depuis broche INT1
IE.1	ET0	interruption TIMER 0
IE.0	EX0	interruption EXTERNE depuis broche INT0

Tableau 12.3 : Registre d'autorisation des interruptions IE

Exemple : Dans un programme on désire prendre en compte les interruptions provenant du port série et de la broche externe INT0.

L'instruction MOV IE,#00010001B

prépare la prise en compte des deux sources choisies, puis

SETB EA

autorise la prise en compte des interruptions alors que

CLR EA

suspend cette prise en compte sans modifier IE.0 et IE.4

Il est recommandé de laisser à 0 les bits inutilisés (bit IE.6 et IE.5 pour le 8051) afin de conserver au programme une compatibilité avec des microcontrôleurs de version 8052 ou autres.

Dans l'exemple cité ci-dessus, deux sources d'interruption sont prises en compte par le microcontrôleur. Il est important de savoir laquelle de ces deux sources serait prise en compte dans le cas où sur le même cycle machine, les deux sources seraient échantillonnées comme active.

Il faut savoir que pour l'instant la mise en place des autorisations des interruptions par le positionnement des bits du registre IE confère un ordre de prise en compte intrinsèque au processeur même si toutes les sources sont considérées d'un même

niveau de priorité. En effet l'échantillonnage des demandes est fait selon une séquence de test définissant une hiérarchie implicite des prises en compte des requêtes d'interruption simultanées. Cette hiérarchie correspond à l'ordre de la table des vecteurs et elle est redonnée ci-dessous :

ordre	source	
1	IE0	plus haute priorité naturelle
2	TF0	
3	IE1	
4	TF1	
5	RI+TI	
6	TF2+EXF2	plus basse priorité

Dans notre exemple où INT0 et le port série émettraient en même temps une demande d'interruption, c'est la source INT0 (indicateur IE0) qui serait servie en premier.

Modification de l'ordre de priorité

Il est possible de modifier le niveau implicite de priorité des six sources d'interruption en attribuant à une ou plusieurs de ces sources un niveau supérieur de prise en compte. Cette modification est réalisable par programmation des bits du registre IP (*Interrupt Priority*). Chacun des bits de ce registre, lorsqu'il est positionné à 1, donne à la source correspondante un niveau de priorité supérieur aux demandes émanant des sources de niveau initial. La figure 12.4 permet de situer le rôle des deux registres de contrôle IE et IP.

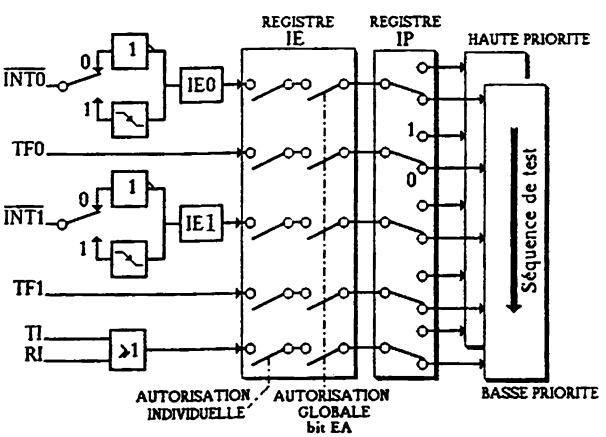


Figure 12.4 : Structure générale du système d'interruption (noyau 8051)

Dans le cas où deux indicateurs d'interruption affectés d'un niveau de priorité supérieur sont actifs en même temps, c'est alors la hiérarchie implicite déterminée par la séquence de test qui détermine l'ordre de prise en compte de ces deux interruptions.

Le tableau 12.5 donne le détail d'affectation du registre IP.

IP : Adresse directe B8H et adressable au niveau du bit

BF	BE	BD	BC	BB	BA	B9	B8
—	—	PT2	PS	PT1	PX1	PT0	PX0

position	symbole	fonction
IP.7	—	réservé à une fonction future
IP.6	—	réservé à une fonction future
IP.5	PT2	priorité interruption TIMER 2 (sur 8052)
IP.4	PS	priorité interruption port série
IP.3	PT1	priorité interruption TIMER 1
IP.2	PX1	priorité interruption externe INT1
IP.1	PT0	priorité interruption TIMER 0
IP.0	PX0	priorité interruption externe INT0

Tableau 12.5 : Registre de priorité des interruptions IP

Prise en compte d'une demande d'interruption

Les indicateurs d'interruption sont échantillonnés durant la période S5P2 de chaque cycle machine. L'échantillon est analysé pendant le cycle machine suivant. Pour le timer 2 spécifique au 8052 la procédure est différente, et son mode de prise en compte est décrit dans le paragraphe "temps de reponse".

Si à l'instant S5P5 d'un cycle machine un des indicateurs est actif, à la fin du cycle suivant (cycle d'analyse) le système d'interruption génère un appel à un sous-programme dont l'adresse est appropriée à la source de chaque indicateur. Cet appel de sous-programme peut être bloqué si l'une des conditions suivantes est détectée :

1 - une interruption de même niveau de priorité ou de niveau supérieur est déjà en cours

2 - le cycle machine correspondant à l'analyse n'est pas le dernier de l'instruction en cours d'exécution

3 - l'instruction en cours d'exécution est un ordre de retour de sous-programme (RETI) ou une instruction correspondant à une écriture dans l'un des registres IE ou IP.

La condition 3 assure l'exécution d'une instruction après modification des registres définissant le mécanisme des interruptions et après le saut correspondant au retour de traitement de l'interruption. C'est seulement après l'exécution de cette instruction supplémentaire qu'une autre requête d'interruption sera prise en compte et traitée.

Dans le cas d'un blocage du processus d'interruption par l'une des trois conditions citées qu'advient-il de la requête non servie ?

Etant donné que l'échantillonnage des indicateurs d'interruption est renouvelé à chaque cycle machine, le nouvel échantillon remplace donc l'ancien et si la demande d'interruption est toujours présente, elle sera prise en compte sur le cycle machine suivant. Dans le cas où la demande d'interruption est issue d'une source interne (port série, timer...) l'indicateur restera positionné car celui-ci est soit inchangé, par principe, soit remis à zéro au moment du saut au sous-programme de service. Donc la demande d'interruption sera prise en compte même après blocage, sauf dans le cas où, par logiciel, l'utilisateur remet de lui même l'indicateur à 0 !

Par contre dans le cas d'une interruption de source externe programmée sur niveau logique, il est possible qu'une demande d'interruption ne soit jamais prise en compte. En effet, sur blocage de demande il faut être sûr que le changement d'état persistera jusqu'au prochain échantillonnage, soit un cycle machine plus tard, pour qu'une nouvelle analyse conduise à l'exécution de la demande.

Lorsque le processeur accepte la requête d'interruption, le saut automatique au sous-programme approprié est exécuté. C'est à ce moment-là qu'éventuellement l'indicateur de demande d'interruption est remis à zéro. Pour plus de précision il est nécessaire d'analyser en détail les particularités de chaque source d'interruption.

Interruptions extérieures INT0 ET INT1

Deux sources d'interruption extérieure sont possibles à partir de deux lignes du port P3. Il s'agit des lignes P3.2 et P3.3 dont les fonctions secondaires sont respectivement INT0 et INT1. Une interruption extérieure peut être définie comme survenant pour un changement d'état logique sur l'une de ces deux lignes (interruption sur niveau) ou sur détection d'un changement d'état (interruption sur front négatif ou descendant), le choix est réalisé par programmation des deux bits IT0 et IT1 situés dans le registre TCON (voir chapitre timer).

Si le bit IT0 (ou IT1) est mis à 1, l'interruption surviendra pour un changement d'état 0 vers 1 (front descendant).

Si le bit IT0 (ou IT1) est mis à 0, l'interruption surviendra pour la présence d'un état logique 0 sur la broche INT0 (ou INT1).

Dans le cas d'une interruption par front la broche d'entrée est échantillonnée pendant la durée d'un cycle machine. Pour que la transition soit correctement prise en compte le niveau 1 et le niveau 0 encadrant la transition doivent donc durer au moins un temps égal à un cycle machine, c'est-à-dire un temps égal à 12 périodes d'horloge (1 microseconde pour un quartz de 12 MHZ).

L'état de ces sources d'interruption est signalé par deux autres bits de ce même registre, il s'agit des bits IE0 et IE1. Ces bits sont mis à 1 lorsque la condition d'interruption est remplie (transition ou changement d'état), alors si et seulement si la source d'interruption est autorisée, le compteur programme est positionné à l'adresse du sous-programme d'interruption de INT0 ou INT1.

Après branchement à cette adresse, le bit IE0 (ou IE1) est remis à 0 automatiquement lorsque l'événement à détecter est un front descendant.

Par contre, si l'événement à détecter est un changement d'état logique, l'indicateur IE0 (ou IE1) reste à 1 et doit être remis à 0 par programmation.

Interruption par les timers

Sources d'interruption timer 0 et 1

Pour ces deux sources, les demandes d'interruption sont signalées par les indicateurs TF0 et TF1 correspondant respectivement aux bits 7 et 5 du registre TCON. Ces indicateurs sont mis à 1 lors d'un débordement de capacité du compteur (exception faite pour le timer 0 fonctionnant en mode 3). Lorsqu'une "*interruption timer*" est honorée l'indicateur (TF0 ou TF1) est remis automatiquement à 0 au moment du saut vers le sous-programme de service (adresse 000BH pour TF0 et 001BH pour TF1)

Timer 2 du 8052

Une interruption du timer 2 est obtenue à partir des deux indicateurs TF2 et EXF2. La prise en compte est faite sur le résultat d'une opération logique OU entre ces deux indicateurs. Il suffit donc qu'un des deux indicateurs passe à l'état 1 pour qu'une demande d'interruption soit reconnue. L'état de ces indicateurs n'est pas modifié au moment du saut à l'adresse 002BH. Cette particularité permet, par programme, d'analyser l'état de chaque indicateur afin d'envisager éventuellement deux traitements différents. Le sous-programme de service devra comporter la ou les instructions de remise à 0 du ou des indicateurs avant retour au programme principal.

Port série

Comme pour le timer 2, les demandes d'interruptions à partir du port série sont signalées par deux indicateurs. Il s'agit de RI qui indique une demande d'interruption sur réception d'un octet et de TI qui signale que le tampon de transmission est libre à l'envoi d'un nouvel octet. Au moment du saut vers l'adresse 0023H ces indicateurs ne sont pas automatiquement remis à 0. Cette opération est laissée au soin de l'utilisateur après analyse de la nature même de l'interruption : réception d'une donnée ou transmission possible. La demande d'interruption est reconnue sur opération logique OU entre les deux indicateurs RI et TI.

Temps de réponse

Du fait qu'une demande d'interruption puisse être bloquée sur condition particulière et vu la nature différente des sources d'interruption, il est évident que le temps de réponse, entre le moment où le phénomène génère la demande d'interruption et le moment où le sous-programme de service est exécuté, ne soit pas constant. Il reste à définir les temps de réponse minimal et maximal pour tirer profit des possibilités du microcontrôleur. Pour cela il est nécessaire de reprendre tous les cas de figure résultant des particularités du mécanisme de service d'une interruption.

En ce qui concerne les sources externes, l'état présent sur les broches INT0 et INT1 est inversé et mémorisé dans les indicateurs IE0 et IE1 durant la période S5P2 de chaque cycle machine.

De façon similaire, les indicateurs correspondant aux sources internes

— EXF2 du timer 2,

— RI et TI du port série,

— TF0 et TF1 correspondant aux timers 0 et 1,

sont positionnés sur S5P2 et traités sur le cycle machine suivant.

En revanche, l'indicateur TF2 du timer 2 est échantillonné comme actif durant la période S5P2 correspondant au cycle pendant lequel le compteur passe en débordement et la prise en compte et le traitement de la demande sont réalisés sur ce même cycle machine.

En conclusion de cette récapitulation, si une demande d'interruption est active et qu'aucune condition de blocage ne s'oppose au service de celle-ci, le processus de branchement au sous-programme de service prendra effet sur le cycle machine suivant l'échantillonnage ayant révélé la demande. L'exécution d'une instruction d'appel à un

sous-programme durant deux cycles machine, il faut donc compter un minimum de trois cycles machine complets entre l'instant où une demande d'interruption est détectée et l'instant où la première instruction du sous-programme de service va être exécutée.

A ce temps de réponse minimum il faut ajouter un délai variable si l'une des trois conditions de blocage s'oppose au service de l'interruption :

1 - Si une autre interruption de même niveau ou de niveau supérieur est en cours d'exécution, le temps de réponse à la nouvelle demande sera fonction de la durée de traitement du sous-programme en cours. Il n'est donc pas possible d'exprimer dans ce cas, un temps de réponse typique à la demande d'interruption.

2 - Si le cycle machine en cours n'est pas le dernier de l'instruction en cours d'exécution, le délai à ajouter au temps de réponse minimum n'excède pas trois cycles machine car les instructions les plus longues (MUL et DIV) ne nécessitent que quatre cycles. Le temps de réponse pour cette condition de blocage est donc au maximum de six cycles machine.

3 - Si l'instruction en cours d'exécution est un "**RETI**" ou une opération d'écriture dans l'un des registres IP ou IE, le délai à ajouter restera égal ou inférieur à cinq cycles machine : un cycle pour achever l'instruction en cours et de un à quatre cycles pour exécuter l'instruction supplémentaire prévue dans cette condition de blocage. En excluant le cas où les interruptions sont imbriquées, le temps de réponse à une demande d'interruption sera toujours supérieur à **trois** cycles machine et restera inférieur à **neuf** cycles machine.

Exemples d'utilisation

Exécution d'un programme en mode pas à pas

La structure du système d'interruption permet d'envisager une procédure d'exécution d'un programme en mode pas à pas.

Il faut se souvenir qu'une demande d'interruption ne peut pas être prise en compte durant le traitement d'une autre demande d'interruption de même niveau de priorité. La prise en compte et l'exécution de la nouvelle demande restent en suspend jusqu'à la fin du traitement en cours. La fin du sous-programme se termine par l'instruction RETI. Cette instruction provoque le retour au programme principal et autorise la prise en compte de la requête en attente. Après la réalisation du saut de retour, une instruction du programme principal est exécutée avant que l'exécution de la demande d'interruption en attente ait lieu.

En partant de ce principe, le constructeur INTEL propose une idée de programme qui permettrait d'exécuter un programme, instruction après instruction, et ce, au rythme d'impulsions électriques appliquées à une broche du microcontrôleur.

L'exemple utilise l'interruption extérieure INT0. Cette interruption correspond aux événements présents sur la broche P3.2. L'interruption INT0 doit être initialisée pour réagir sur état logique.

```
JNB  P3.2,$   ;attendre ici que la broche P3.2 soit à 1
JB   P3.2,$   ;attendre ici que P3.2 repasse à 0
RETI          ;et retourner exécuter une instruction
```

Cette suite de trois instructions doit être placée à partir de l'adresse 0003. La broche P3.2 est maintenue dans l'état logique 0 ce qui correspond à une demande permanente d'interruption. Le saut à l'adresse 0003 est donc assuré et l'indicateur IE0 correspondant à la source INT0 est automatiquement remis à 0. La première instruction assure une boucle d'attente sur la condition P3.2 = 1. Lorsque le niveau électrique présent sur P3.2 devient proche de 5 volts c'est la deuxième instruction qui assure une attente d'un nouvel état 0. Dès que la broche P3.2 repasse à 0, l'indicateur IE0 est positionné à 1 et donc une nouvelle demande d'interruption est enregistrée. Cette demande ne peut pas être servie avant l'exécution de l'instruction qui suivra l'instruction RETI. Une instruction du programme principal sera donc exécutée puis un saut à l'adresse 0003 assurera une synchronisation à l'impulsion électrique extérieure.

Cet exemple nécessite un circuit extérieur capable de générer des impulsions électriques (oscillateur ou bouton poussoir avec circuit antirebonds). Il est possible d'envisager une variante de ce programme.

La broche P3.2 est maintenue à 1 d'une façon définitive (état naturel d'un port d'entrée laissé en l'air), et l'on agit directement, par logiciel, sur l'indicateur IE0 :

```
LCALL  AFF_REG ; affiche état processeur par exemple
SETB   IE0     ; réamorce l'interruption
RETI           ; et retourne exécuter une instruction.
```

Gestion du port série sous interruption

L'amorce de programme présenté ci-dessous présente une gestion sous interruption du port série. Deux zones de la mémoire interne, de trente deux octets chacune, sont utilisées comme tampon (buffer) de réception et de transmission. Utiliser la communication revient alors à prélever, depuis le tampon de réception, les caractères reçus, ou à déposer les caractères à envoyer dans le tampon de transmission. Le sous-

programme d'interruption prend en charge la gestion des zones tampon. Le programme principal n'est pas très utile et sa simplicité peut rendre discutable la mise en œuvre d'une communication série gérée sous interruption. Il faut quand même remarquer que le programme assure l'envoi des chaînes de caractères **message, retour0 et retour1**. La gestion d'envoi de ces suites de caractères "occupe" le processeur. Durant ces périodes d'envoi, l'arrivée d'informations dans le registre de réception (SBUF) pourrait être perdue par phénomène d'écrasement (une première donnée étant remplacée par une autre avant même d'être prise en compte). La réception sous interruption évite ce risque. D'autre part la transmission sous interruption limite l'attente entre l'envoi de chaque donnée.

La gestion des zones tampon est partagée entre le sous-programme d'interruption et le programme principal. Chaque tampon est géré par deux pointeurs, l'un pour l'écriture, l'autre pour la lecture. La zone de mémoire interne, choisie pour recevoir la valeur de ces pointeurs, correspond à la deuxième banque de registres (banque 1). Les symboles définis correspondent en fait à un moyen d'adresser en mode direct ces registres sans que la banque soit obligatoirement active. Ainsi TXM_RD_PTR correspond à R1 et RCV_WR_PTR à R0 de la banque 1.

Etude du principe de gestion du tampon de réception :

Ce tampon occupe 32 octets à partir de l'adresse "**RCV_BUF**". Au départ les deux pointeurs RCV_WR_PTR et RCV_RD_PTR sont initialisés à la valeur de cette adresse, et le compteur RCV_COUNT prend la valeur 0. Lorsqu'un caractère est reçu, il est placé dans le tampon à l'adresse pointée par RCV_WR_PTR et le compteur RCV_COUNT est incrémenté de 1. Le pointeur RCV_WR_PTR est lui aussi incrémenté de 1 pour la mise en place du prochain caractère. La valeur du pointeur est testée afin de vérifier que la fin du tampon (TXM_BUF+B_LNG) n'est pas atteinte. Si cette extrémité est atteinte, le pointeur est replacé en début de tampon. Les nouvelles données vont donc recouvrir les anciennes, c'est-à-dire que le premier caractère reçu sera remplacé par le trente troisième. Mais entre temps, le programme principal a sans doute prélevé des caractères de ce tampon. Il les prélève grâce au pointeur RCV_RD_PTR depuis le sous-programme RS_IN. Ce sous-programme décrémente le compteur RCV_COUNT et déplace son pointeur de lecture RCV_RD_PTR. Bien que non prévu dans l'exemple, il est donc très facile de lever le doute sur le fait qu'un nouveau caractère en recouvre un précédent sans que ce dernier ai été récupéré par le programme principal : il suffit de tester la valeur du compteur RCV_COUNT et de s'assurer que ce compteur ne dépasse pas la valeur B_LNG (c'est-à-dire 32). Ce test peut être réalisé aussi bien dans le sous-programme RS_IN que dans le sous-programme d'interruption. Il faut noter que le sous-programme RS_IN peut subir d'autres modifications. Pour l'instant le programme principal appelle ce sous-programme sans savoir si un caractère est réellement arrivé dans le tampon de réception. Il y a donc blocage dans le sous-programme avec attente d'une réception.

Si l'application le nécessite, il est possible d'extraire le test de RCV_COUNT (deux premières lignes de RS_IN) et de faire ce test depuis le programme principal. Le résultat de ce test permet alors le choix entre récupérer un caractère ou faire autre chose...

Les directives d'assemblage DSEG, DS, BSEG, DBIT sont reconnues par l'assembleur INTEL ASM51. Elles sont décrites dans le chapitre 14.

```
B_LNG       EQU   32        ;LONGUEUR DES TAMPONS (BUFFER)
BANK_1      EQU   00001000B
```

LISTE DES VARIABLES PLACEES EN RAM INTERNE

```
            DSEG            AT 08H
;POINTEUR POUR ECRITURE DANS BUFFER DE RECEPTION = R0 BANQUE 1
RCV_WR_PTR: DS    1         ;à partir de l'adresse 08H
;POINTEUR POUR LECTURE DANS BUFFER DE TRANSMISSION
TXM_RD_PTR: DS    1         ;correspond à R1 BANQUE 1
;POINTEUR POUR LECTURE DANS BUFFER DE RECEPTION
RCV_RD_PTR: DS    1
;POINTEUR POUR ECRITURE DANS BUFFER DE TRANSMISSION
TXM_WR_PTR: DS    1
;COMPTEUR DE CARACTERES RECUS
RCV_COUNT:  DS    1
;COMPTEUR DE CARACTERES à ENVOYER
TXM_COUNT:  DS    1
```

ZONES DE MEMOIRE TAMPON
EN RAM INTERNE A PARTIR DE 30H

```
            DSEG  AT 30H
RCV_BUF:    DS    B_LNG       ;BUFFER DE RECEPTION
```

```
TXM_BUF :  DS      B_LNG        ;BUFFER DE TRANSMISSION
STACK :    DS      1            ;DEBUT DE LA PILE
```

LISTE DES VARIABLES "BIT" *

```
           BSEG    AT (0)
FLAG_TXM : DBIT    1            ;=1 ENVOI AMORCE
```

DEBUT DU CODE EXECUTABLE

```
           CSEG    AT 0         ; ADDRESSE RESET
           ORG     0H
FROID :
           LJMP    INI_SYSTEME  ;APRES RESET SAUT A INI_SYSTEME
```

SOUS-PROGRAMME D'INTERRUPTION DU PORT SERIE
PLACE A L'ADRESSE 23H
IL FAUT TESTER LES INDICATEURS TI ET RI POUR
IDENTIFIER LA CAUSE EXACTE DE L'INTERRUPTION

```
       ORG     23H              ; VECTEUR INTERRUPTION
SERIAL_INT:                     ; POUR PORT SERIE
       PUSH    PSW
       PUSH    ACC
       MOV     PSW,#BANK_1      ; activer banque de registres 1
       JBC     RI,RX_INT        ; SI RI=1 SAUT ET RI=0
TX_INT :
       MOV     A,TXM_COUNT      ; GESTION DE L'ENVOI :
       JNZ     TX_INT0          ; un caractère est a envoyer ?
```

```
        CLR     FLAG_TXM        ; non alors désactiver indicateur
        SJMP    TX_IT2          ; et sortir
TX_INT0 :
        MOV     SBUF,@R1        ; OUI : envoyer car. vers SBUF
        INC     R1              ; mise à jour pointeur
        DEC     TXM_COUNT       ; et COMPTEUR
        CJNE    R1,#TXM_BUF+B_LNG,TX_IT2
        MOV     R1,#TXM_BUF     ; réinitialisation pointeur
TX_IT2: CLR     TI
        SJMP    IT_END
; L'interruption est due à un caractère reçu
RX_INT :
        MOV     @R0,SBUF        ; SI RI=1 placer caractère
        INC     R0              ; mise à jour pointeur
        INC     RCV_COUNT       ; et compteur de caractères
        CJNE    R0,#RCV_BUF+B_LNG,IT_END
        MOV     R0,#RCV_BUF     ; init pointeur
IT_END :
        POP     ACC
        POP     PSW             ; réactive BANQUE REG. 0
        RETI
```

ENTREE APRES RESET :
INITIALISATION DES VARIABLES, DE LA COMMUNICATION SERIE
ET AUTORISATION INTERRUPTION DEPUIS PORT SERIE

```
INI_SYSTEME :
        MOV     SP,#STACK
        MOV     RCV_WR_PTR,#RCV_BUF ; POINTEURS SUR DEBUT
        MOV     RCV_RD_PTR,#RCV_BUF ; DES BUFFERS
        MOV     TXM_WR_PTR,#TXM_BUF
        MOV     TXM_RD_PTR,#TXM_BUF
```

```
        MOV     TXM_COUNT,#0 ; COMPTEURS A 0
        MOV     RCV_COUNT,#0
        CLR     FLAG_TXM
```

INITIALISATION COMMUNICATION SERIE
QUARTZ DE 11.059 MHZ
LE TIMER 1 EST UTILISE COMME HORLOGE
LA VITESSE DE COMMUNICATION EST FIXEE A 9600 BAUDS

```
        MOV     SCON,#01011010B ;TI=1 POUR PREMIER ENVOI
        MOV     TMOD,#20H       ;8 BIT AUTO RELOAD TIMER 1
        MOV     TH1,#0FDH       ;VALEUR POUR 9600 BAUDS
        SETB    TR1             ;HORLOGE ACTIVE
```

MISE EN SERVICE DES INTERRUPTIONS POUR PORT SERIE

```
;VALIDE INT GENERALE ET PORT SERIE
        MOV     IE,#10010000B
```

PROGRAMME PRINCIPAL

```
MAIN :  MOV     DPTR,#MESSAGE   ; sortir un message
        CALL    OUT_STRG
MAIN_0 :
        CALL    RS_IN           ; attente d'un caractère
        CALL    RS_OUT          ; écho sur sortie série
        CJNE    A,#'1',PAS_UN   ; le caractère reçu correspond
        MOV     DPTR,#RETOUR1   ; au code ASCII de 1
        CALL    OUT_STRG        ; alors envoi d'un message
        SJMP    MAIN_0          ; et refaire
```

```
PAS_UN :  CJNE   A,#'0',PAS_0
          MOV    DPTR,#RETOUR0
          CALL   OUT_STRG
PAS_0 :   ........                ; suite du programme
          LJMP   MAIN_0           ; LOOP
```

SORTIE VERS PORT SERIE D'UNE CHAINE
POINTEE PAR DPTR ET TERMINEE PAR LE CODE 0

```
OUT_STRG :
          CLR    A
          MOVC   A,@A+DPTR
          CJNE   A,#0,MES1
          RET
MES1 :    CALL   RS_OUT
          INC    DPTR
          SJMP   OUT_STRG
```

ATTENTE D'UN CARACTERE DEPUIS BUFFER RECEPTION

```
RS_IN :   MOV    A,RCV_COUNT    ; test du compteur RCV_COUNT
          JZ     RS_IN          ; attendre ici si pas de caractère
RS_IN0 :
          DEC    RCV_COUNT
          MOV    R1,RCV_RD_PTR
          MOV    A,@R1
          INC    R1
          CJNE   R1,#RCV_BUF+B_LNG,RS_IN1
          MOV    R1,#RCV_BUF
RS_IN1 :
          MOV    RCV_RD_PTR,R1
          RET
```

DEPOT D'UN CARACTERE DANS BUFFER DE TRANSMISSION

```
RS_OUT:
        MOV   B,A                          ; sauvegarde caractère à envoyer
RS_OUT3:
        MOV   A,TXM_COUNT
        CJNE  A,#B_LNG-1,RS_OUT2           ; BUFFER plein ?
RS_OUT2:
        JNC   RS_OUT3                      ; ALORS attendre
        MOV                                ; R1, TXM_WR_PTR ; SINON
        MOV   @R1,B                        ; mise en place du caractère
        INC   R1                           ; dans buffer et mise à jour
        INC   TXM_COUNT                    ; nombre caractères en attente
        CJNE  R1, # TXM_BUF+B_LNG,RS_OUT0
        MOV   R1, # TXM_BUF                ; mise à jour pointeur
RS_OUT0:
        MOV   TXM_WR_PTR,R1
        MOV   A,B
        JB    FLAG_TXM,RS_OUT1             ; saut si envoi déjà amorcé
        SETB  FLAG_TXM
        CLR   EA                           ; provoquer interruption
        SETB  TI                           ; pour envoi premier
        SETB  EA                           ; caractère
RS_OUT1:
        RET
;LISTE DES MESSAGES
MESSAGE:        DB ' SYSTEME PRET!!! ',0
RETOUR1:        DB ' VALEUR 1 ',13,10,0
RETOUR0:        DB ' VALEUR 0 ',13,10,0
                END
```

13

Processeur Booléen

Définition du terme "processeur booléen"

Le terme de processeur booléen a été évoqué dans les chapitres 2 et 6. Rappelons simplement que cette expression désigne la possibilité d'utiliser l'unité centrale 8 bits comme une unité de traitement au niveau du bit. Il est donc possible de tester une entrée, de positionner une sortie sans modifier le reste de l'ensemble 8 bits formé par un port. Cet adressage au niveau du bit caractérise aussi une partie de la mémoire interne. Ce type d'adressage est exploité par une partie du jeu d'instructions (tableaux 13.1a et 13.1b). Le processeur booléen utilise comme "registre accumulateur" l'indicateur de report C.

Mnémonique	Opération	Cycle
ANL C, bit	C=C and bit	2
ANL C,/bit	C=C and not bit	2
ORL C, bit	C=C or bit	2
ORL C,/bit	C=C or not bit	2
MOV C,bit	C=bit	1
MOV bit,C	bit=C	2
CLR C	C=0	1
CLR bit	bit=0	1
SETB C	C=1	1
SETB bit	bit=1	1
CPL C	C=not C	1
CPL bit	bit=not bit	1

Tableau 13.1a : Instructions boléennes

Mnémonique	Opération	Cycle
JC REL	saut si C=1	2
JNC rel	saut si C=0	2
JB bit, rel	saut si bit=1	2
JNB bit,rel	saut si bit=0	2
JBC bit,rel	saut si bit=1 & bit=0	2

Tableau 13.1 b : Sauts conditionnels utilisant l'adressage au niveau du bit

Intérêt de l'utilisation du processeur booléen

Afin d'apprécier l'utilité du processeur booléen, on se propose de traiter un exemple très simple résumé par l'équation logique :

$$S = \bar{e}$$

où S représente la broche P1.0, et e la broche P1.1

Dans une approche classique (sans utiliser le processeur booléen), ce problème simple doit se décomposer de la façon suivante :

1. Lire l'ensemble 8 bits du port P1 ;
2. isoler l'état de la broche P1.1 ;
3. tester l'état du bit isolé pour définir l'état de S ;
4. replacer sur le port cet état sans modifier le reste du port

Ce qui se traduit par la suite d'instructions

```
        MOV  A,P1          ; ETAT DU PORT DANS ACCUMULATEUR
        ANL  A,#00000010B  ; MASQUE POUR P1.1
        JZ   E_ZERO        ; SI A=0 E=0
        ANL  P1,#11111110B ; SINON E=1 D'OU S=0
        SJMP SORTIE
E_ZERO: ORL  P1,#00000001B ; E=0 DONC S=1
SORTIE: .....
```

L'utilisation du processeur booléen permet de réduire considérablement le traitement de cet exemple :

```
MOV     C,P1.1          ; LIRE L'ENTREE
CPL     C               ; COMPLEMENTER L'ETAT
MOV     P1.0,C          ; ACTUALISER LA SORTIE
```

On remarquera que la première solution utilise la possibilité de réaliser des opérations logiques directement sur un registre de port. Cette façon de faire est rarement possible sur un microprocesseur classique. Un transit par l'accumulateur est alors nécessaire ce qui alourdit encore plus le traitement du problème.

Programmation d'un circuit de logique combinatoire

L'équation logique

$$Q = \left(u\,(v + w)\right) + (x.\overline{y}) + \overline{z}.$$

peut se traduire par le logigramme de la figure 13.2. D'un point de vue technologique, il est possible de réaliser une telle fonction à partir de circuits intégrés (TTL ou CMOS) ou bien à partir d'un ensemble de contacts et de relais (figure 13.3.). Bien que facultatifs, les relais auxiliaires KA1 et KA2 représentent des résultats intermédiaires du calcul. Leur utilisation simplifient la compréhension du fonctionnement.

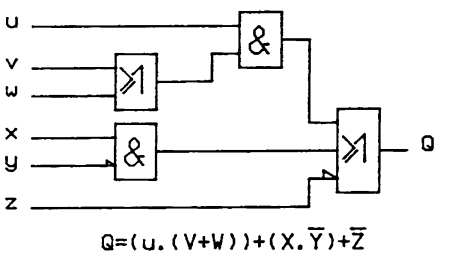


Figure 13.2 : Logigramme du circuit à programmer

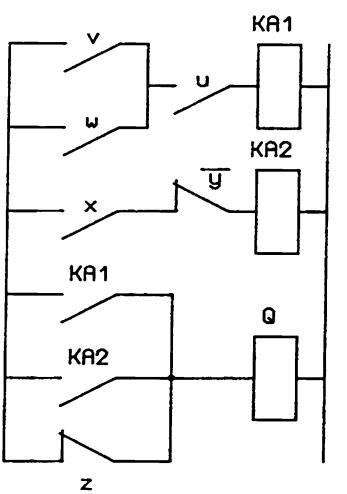
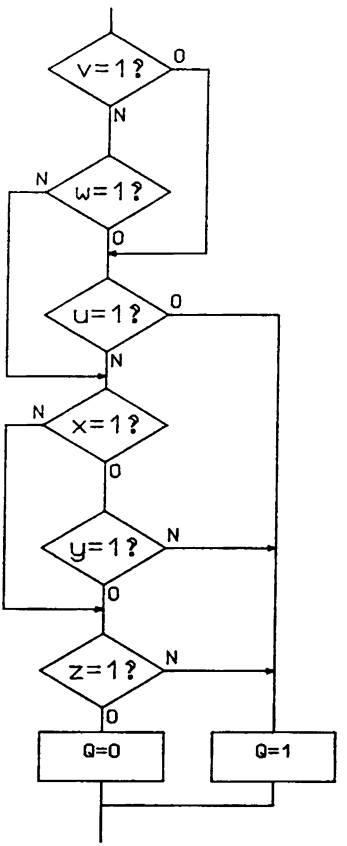


Figure 13.3 : Schéma à relais

Il est aussi possible d'envisager de remplir cette même fonction par programmation à partir d'un microcontrôleur. Une approche classique de programmation conduit à établir un organigramme tel que celui de la figure 13.4.



u	P1.1	bit 1 du port P1
v	p2.2	bit 2 du port P2
w	TF0	indicateur de débordement TIMER 0
x	IE0	interruption externe 0
y	20H.0	bit 0 de l'octet d'adresse 20H
z	21H.1	bit 1 de l'octet 21H
Q	P3.3	bit 3 du port P3

Figure 13.4 : Organigramme

Tableau 13.5 : Affectation entrées et sorties

Le jeu d'instructions du 8051 permet d'envisager trois façons de résoudre ce problème. La première de ces solutions est inspirée de la méthode utilisée sur des processeurs classiques, les deux autres utilisent les possibilités offertes par le processeur booléen et son mode d'adressage au niveau du bit.

L'affectation des entrées et de la sortie (tableau 13.5) reste la même dans les trois exemples. Elle est choisie arbitrairement.

Solution utilisant un adressage au niveau de l'octet

Le test de l'état d'une variable d'entrée est réalisé par opération de masquage et un saut est exécuté selon le résultat de cette opération (nul ou non nul). Les différents masques sont établis en fonction de la position de la variable d'entrée dans l'octet auquel elle appartient. Le programme respecte l'organigramme de la figure 13.4.

```
SORTIE   EQU   22H            ; représente l'état des 8 bits de P3
TST_V:   MOV   A,P2           ; lecture de V
         ANL   A,#00000100B   ; masque pour isoler P2.2
         JNZ   TST_U          ; V=1 alors tester u
         MOV   A,TCON         ; lire w (TF0)
         ANL   A,#00100000B   ; isoler TF0 dans TCON
         JZ    TST_X          ; si w= 0 alors tester X
TST_U:   MOV   A,P1           ; lecture de u dans P1
         ANL   A,#00000010B   ; isoler u=P1.1
         JNZ   SET_Q          ; si u=1 Q=1 et fin
TST_X:   MOV   A,TCON         ; lire x dans TCON
         ANL   A,#00001000B   ; isoler x=IE0
         JZ    TST_Z          ; si x=0 alors tester z
         MOV   A,20H          ; sinon lire y dans 20H
         ANL   A,#00000001B   ;       isoler y
         JZ    SET_Q          ; si y=0 faire Q=1
TST_Z:   MOV   A,21H          ; sinon tester z
         ANL   A,#00000010B
         JZ    SET_Q          ; si z=0 faire Q=1
CLR_Q:   MOV   A,SORTIE       ; mettre Q à 0 à partir
         ANL   A,#11110111B   ; de SORTIE
         SJMP  OUT_Q
SET_Q:   MOV   A,SORTIE       ; mise à 1 de Q
         ORL   A,00001000B
OUT_Q:   MOV   SORTIE,A       ; mise à jour de SORTIE
         MOV   P3,A           ; et du port P3
```

Solution utilisant les instructions de test de bit

Ce deuxième programme utilise les instructions du tableau 13.1b. Sa conception découle directement de l'organigramme. Il offre donc une bonne lisibilité. De plus, l'absence de l'emploi de masque servant à isoler les variables permet de limiter les risques d'erreurs d'écriture. On passe de 25 lignes de programme à 9 lignes pour le même résultat !

```
TST_V :  JB    V,TST_U     ; si V=1 tester U
         JNB   W,TST_X     ; sinon si W=0 tester X
TST_U :  JB    U,SET_Q     ; si U=1 faire Q=1.
TST_X :  JNB   X,TST_Z     ; si X=0 tester Z
         JNB   Y,SET_Q     ; sinon si Y=0 faire Q=1.
TST_Z :  JNB   Z,SET_Q     ; si Z=0 alors faire Q=1
CLR_Q :  CLR   Q           ; sinon Q=0
         SJMP  SUITE       ; et fin
SET_Q :  SETB  Q           ;
SUITE : .....
```

Le temps d'exécution est variable : il dépend de l'état des variables d'entrées. Il est au minimum de 7 cycles machine et au maximum de 13 cycles machine.

Utilisation des instructions booléennes

Cette troisième version utilise les instructions du tableau 13.1a. Le temps d'exécution est constant (16 cycles machine) car aucun saut conditionnel n'est utilisé. Si les deux premières versions prenaient l'organigramme comme modèle, cette version est calquée sur l'écriture de l'équation logique.

On remarquera l'utilisation du bit F0 du registre PSW utilisé comme mémoire d'un résultat partiel. Cette utilisation peut être comparée à l'utilisation des relais auxiliaires KA1 et KA2 dans le schéma de la figure 13.3.

```
MOV   C,v      ;
ORL   C,w      ; OU logique entre v et w
ANL   C,u      ; C = u(v+w)
MOV   F0,C     ; ce résultat est placé dans F0.
```

```
MOV   C,x      ;
ANL   C,/y     ; ET logique entre x et complément de y
ORL   C,F0     ;
ORL   C,/z     ; traitement du OU à 3 entrées
MOV   Q,C      ; résultat vers sortie Q
```

Dans le cas où la rapidité de traitement n'est pas une exigence fondamentale, la conception d'un tel programme reste aisée même pour des équations logiques complexes : il suffit de généraliser l'emploi de bits mémoires afin de stocker des résultats intermédiaires. Si le temps semble être compté, il faut quand même avoir présent à l'esprit qu'une réalisation basée sur une technologie à relais électromagnétique est loin d'être instantanée ! Le temps de collage d'un très bon relais reste supérieur à la milliseconde et un capteur d'information, pour peu qu'il soit doté de part son principe, d'un contact mécanique, possédera un temps de réponse non négligeable.

L'exemple qui suit ne cherche donc pas l'optimisation même si celle-ci est parfois évidente. Le programme est écrit d'une façon systématique comme pourrait le faire un compilateur chargé de traduire une équation logique en code source.

Soit à réaliser l'équation logique

$$S = a\left(b + \overline{c}\,(v + x)\right) + e\,(y + \overline{z})$$

L'ordre de résolution doit prendre en compte le degré de mise en facteur. On commence par le degré de parenthèse le plus profond.

où t1 représente une mémoire temporaire

$$S = a\left(b + \overline{c}\,(t1)\right) + e\,(y + \overline{z})$$

puis faire $t1 = \overline{c}\,.\,t1$

pour arriver à $S = a\,(b + t1) + e\,(y + \overline{z})$

etc. pour arriver finalement à $S = t1 + t2$.

t1 et *t2* sont deux bits situés dans l'octet de RAM interne d'adresse 20H (début de la zone adressable au niveau du bit).

```
t1      bit 0          ;bit 0 de l'octet 20H
t2      bit 1          ;bit 1 de l'octet 20H
        MOV   C,x
```

```
ORL     C,v
MOV     t1,C        ; t1 = x+v
MOV     C,t1
ANL     C,/c
MOV     t1,C        ; t1 = /c(x+v)
MOV     C,t1
ORL     C,b
MOV     t1,C        ; t1 = a+/c(x+v)
MOV     C,y         ; OU logique entre y
ORL     C,/z        ; et le complément de z
MOV     t2,C        ; placé dans t2
MOV     C,t1
ANL     C,a
MOV     t1,C        ; t1 = a . t1
MOV     C,t2
ANL     C,e
MOV     t2,C        ; t2 = e . t2
MOV     C,t1
ORL     C,t2
MOV     S,C         ; définir l'état de sortie
```

Cette suite d'instructions peut bien sûr être simplifiée. Il est inutile de laisser les suites

```
MOV     t1,C
MOV     C,t1
```

Programmation d'un circuit de logique séquentielle

La représentation du type GRAFCET de niveau 2 va être utilisée pour décrire les exemples à traiter. Le GRAphe Fonctionnel de Commande Etape Transition constitue un modèle graphique basé sur les notions d'étape et de réceptivité associées aux transitions. Une étape constitue une situation invariable de la commande de l'automatisme. Les actions associées à une étape ne sont assurées que si l'étape est dite "active". Le grafcet est constitué d'une alternance obligatoire d'étapes et de transitions.

Une transition ne peut être valide que si toutes les étapes, immédiatement précédentes qui lui sont reliées, sont actives.

Une transition est franchissable si la transition est valide et que la réceptivité qui lui est associée est vraie.

Une réceptivité s'exprime sous forme d'une équation logique précisant les relations de prise en compte des entrées de la commande.

Le franchissement d'une transition provoque simultanément :

— la désactivation de toutes les étapes immédiatement précédentes reliées à cette transition ;

— l'activation de toutes les étapes immédiatement suivantes reliées à cette transition.

Dernier rappel : Pour qu'un grafcet puisse évoluer normalement, il est nécessaire qu'au moins une étape soit active à l'initialisation. C'est en général le rôle de l'étape initiale représentée par une étape dont les côtés sont doublés (voir l'étape 0 du premier exemple).

Programmation d'une séquence simple

L'exemple de la figure 13.6 présente une séquence simple de trois étapes (0, 1 et 2). Ces trois étapes sont séparées par trois transitions auxquelles sont associées les réceptivités a(b+c), x et y. L'étape 1 provoque l'activation des sorties EV+ et M, l'étape 2 active les sorties EV— et R. Il est sous-entendu que toute sortie non activée est forcée à 0.

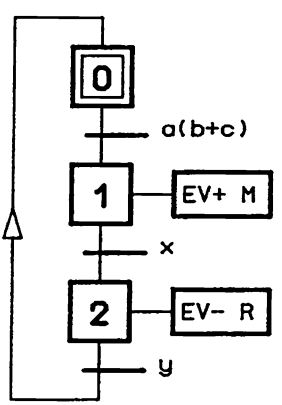


Figure 13.6 : séquence de 3 étapes.

La conception générale du programme sera organisée selon quatre phases :

1. lire les entrées ;
2. définir l'état des réceptivités ;
3. actualiser l'état des étapes ;
4. actualiser l'état des sorties.

Au niveau programmation, des bits de la RAM interne vont être associés aux étapes, et aux réceptivités. Ces bits sont situés dans la zone de mémoire interne adressable au niveau du bit.

Le programme qui suit est donné comme principe de base. Il ne tient pas compte de certaines sécurités à prévoir :

— réinitialisation après coupure d'alimentation ;

— fiabilité de l'actualisation des sorties ;

— filtre des états parasites pouvant être lus sur les entrées ;

— surveillance du bon déroulement du programme ;

etc .

BIT MAP départ octet adresse 20H

```
            BSEG AT (0)
;
ETAPE0 :    DBIT 1      ; réserve 1 bit pour étape 0
ETAPE1 :    DBIT 1      ; réserve pour étape 1
ETAPE2 :    DBIT 1      ; réserve pour étape 2
            DBIT 1      ; réserver le reste de l'octet
            DBIT 1      ; pour éventuelle extension
;
; laisser libre la fin de l'octet 20H
; et passer à l'octet 21H
            BSEG AT (8)
```

```
RECEP1 :    DBIT 1                  ; résultat réceptivité a(b+c)
RECEP2 :    DBIT 1                  ; réceptivité x
RECEP3 :    DBIT 1                  ; réceptivité y
            DBIT 1                  ; pour extension
            DBIT 1
            BSEG    AT (0FH)
; OCTET 22H = IMAGE du port P1
; toutes les variables d'entrée sont câblées sur P1
; tel que a=P1.0 b=P1.1 c=P1.2 x=P1.3 et y=P1.4
;
IN_a :      DBIT 1
IN_b :      DBIT 1
IN_c :      DBIT 1
IN_x :      DBIT 1
IN_y :      DBIT 1
            DBIT 1
;Segment code à partir de l'adresse 0 dans EPROM
            CSEG    AT (0)
;
INIT :      MOV     20H,#01     ; seule l'étape 0 est active
            MOV     21H,#0      ; toutes les réceptivités à 0
;lecture des entrées
RD_IN :     MOV     22H,P1      ; copie du port P1 dans RAM.
;calcul des
réceptivités
;
            MOV     C,IN_c      ; faire c+b
            ORL     C,IN_b      ; C=c+b
            ANL     C,IN_a      ; C=a(b+c)
            MOV     RECEP1,C    ; ranger réceptivité a(b+c)
;
            MOV     C,IN_x      ; idem pour réceptivité 2
            MOV     RECEP2,C
```

```
        MOV     C,IN_y      ; idem pour réceptivité 3
        MOV     RECEP3,C

;Actualiser les bits ETAPE
        MOV     C,ETAPE0    ; si ETAPE0 et RECEP1 =1
        ANL     C,RECEP1    ; alors ETAPE0=0 et ETAPE1=1
        MOV     ETAPE1,C    ; sinon laisser dans l'état
        CPL     C
        MOV     ETAPE0,C
; étape 1
        MOV     C,ETAPE1    ; si ETAPE1 et RECEP2 =1
        ANL     C,RECEP2    ; alors ETAPE1=0 et ETAPE2=1
        MOV     ETAPE2,C    ; sinon laisser dans l'état
        CPL     C
        MOV     ETAPE1,C    ;
; étape 2
        MOV     C,ETAPE2    ; si ETAPE2 et RECEP3 =1
        ANL     C,RECEP3    ; alors ETAPE2=0 et ETAPE0=1
        MOV     ETAPE0,C    ; sinon laisser dans l'état
        CPL     C
        MOV     ETAPE2,C    ; le "rebouclage" est réalisé
;Actualiser les sorties à partir des bits "ETAPE"
        JNB     ETAPE0,OUT1
        CLR     P3.0        ; P3.0 = EV— = 0
        CLR     P3.1        ; P3.1 = R = 0
        CLR     P3.2        ; P3.2 = EV+ à 0 par sécurité
        CLR     P3.3        ; P3.3 = M à 0 par sécurité
OUT1 :  JNB     ETAPE1,OUT2
        SETB    P3.2        ; P3.2 = EV+
        SETB    P3.3        ; P3.3 = M
```

```
OUT2 :    JNB     ETAPE2,FIN
          CLR     P3.2        ; si ETAPE2
          CLR     P3.3        ; RAZ sortie EV+ et M
          SETB    P3.0        ; puis EV— et R = 1
          SETB    P3.1
; Puis recommencer les quatre phases
FIN :     JMP     RD_IN
;
```

Divergence et convergence en ET

Le grafcet de la figure 13.7, bien qu'incomplet (aucune sortie n'est affectée aux 7 étapes), présente l'utilisation d'une divergence et d'une convergence en ET. La divergence en ET se traduit par la règle suivante :

Si l'étape 0 est active et si la réceptivité a est vraie alors l'étape 0 est désactivée et les deux étapes 1 et 4 sont activées.

Puis les deux branches évoluent indépendamment l'une de l'autre. Les étapes 3 et 5 constituent des étapes d'attente. Pour que l'étape 6 soit activée, il faut que les étapes 3 et 5 soient actives et que la réceptivité x soit vraie.

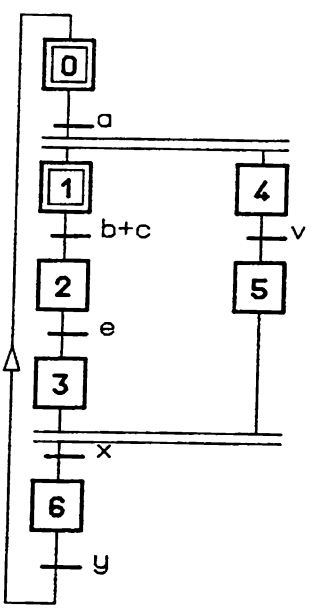


Figure 13.7

L'actualisation des bits "ETAPE" de ce grafcet est donnée ci-dessous. On remarquera que l'ordre de traitement n'a pas une importance primordiale. On peut choisir par exemple un traitement ordonné par branche en adoptant une "priorité" de la gauche vers la droite.

```
.....
;Actualiser les bits ETAPE
;traitement de la divergence en ET
            MOV   C,ETAPE0   ; si ETAPE0 et RECEP1 =1
            ANL   C,RECEP1   ; RECEP1=a
            MOV   ETAPE1,C   ; et ETAPE1=1
            MOV   ETAPE4,C   ; et ETAPE4=1
            CPL   C
            MOV   ETAPE0,C   ; et ETAPE0=0 ; activer étape 2 ?
            MOV   C,ETAPE1   ; si ETAPE1 et RECEP2 =1
            ANL   C,RECEP2   ; RECEP2=b+c
            MOV   ETAPE2,C   ;
            CPL   C
            MOV   ETAPE1,C   ;
; activer étape 3 ?
            MOV   C,ETAPE2   ; si ETAPE2 et RECEP3 =1
            ANL   C,RECEP3   ; RECEP3=e
            MOV   ETAPE3,C   ;
            CPL   C
            MOV   ETAPE2,C   ;
; activer étape 5 ?
            MOV   C,ETAPE4   ; si ETAPE4 et RECEP4 =1
            ANL   C,RECEP4   ; RECEP4=v
            MOV   ETAPE5,C   ;
            CPL   C
            MOV   ETAPE4,C   ;
; convergence en ET activer étape 6 ?
            MOV   C,ETAPE3   ; si ETAPE3
            ANL   C,ETAPE5   ; et si ETAPE5
            ANL   C,RECEP5   ; et si x=1
            MOV   ETAPE6,C   ;
            CPL   C
```

```
            MOV   ETAPE3,C    ; désactiver ETAPE3
            MOV   ETAPE5,C    ; et ETAPE5
; activer étape 0 ?
            MOV   C,ETAPE6    ; si ETAPE6
            ANL   C,RECEP6    ; et si y=1
            MOV   ETAPE0,C    ; ETAPE0=1
            CPL   C           ;
            MOV   ETAPE6,C    ; ETAPE6=0
```

Divergence et convergence en OU

Une divergence en OU traduit la notion de choix d'une séquence d'opérations parmi d'autres séquences. La figure 13.8 présente une divergence en OU entre la séquence ETAPE 1, ETAPE 2 et la séquence ETAPE 3, ETAPE 4. Si une séquence est exécutée, l'autre ne le sera pas.

Si l'étape 0 est active et que la réceptivité a est vraie, alors : l'étape 0 est désactivée et l'étape 1 est activée.

Si l'étape 0 est active et que la réceptivité b est vraie, alors :

l'étape 0 est désactivée et l'étape 3 est activée.

On observe que la désactivation de l'étape 0 implique qu'une seule des deux conditions pourra être remplie.

La vérification de ces 2 conditions ne pouvant être simultanée, l'ordre du traitement prend ici une grande importance. Il est nécessaire de fixer une priorité dans le cas où les réceptivités a et b seraient vraies en même temps. Dans le programme qui suit, la priorité donnée est celle d'une lecture du grafcet allant de gauche vers la droite.

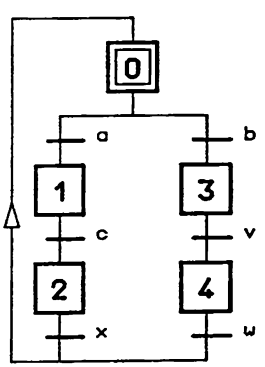


Figure 13.8

```
;traitement de la divergence en OU
        MOV   C,ETAPE0   ; si ETAPE0 et RECEP1 =1
        ANL   C,RECEP1   ; RECEP1=a
        MOV   ETAPE1,C   ; et ETAPE1=1
        CPL   C
        MOV   ETAPE0,C   ; et ETAPE0=0
; activer étape 2 ?
        MOV   C,ETAPE1   ; si ETAPE1 et RECEP3 =1
        ANL   C,RECEP3   ; RECEP3=c
        MOV   ETAPE2,C   ;
        CPL   C
        MOV   ETAPE1,C   ;
; activer étape 3 dans le cas où a= 0
        MOV   C,ETAPE0   ; si ETAPE0 et RECEP2 =1
        ANL   C,RECEP2   ; RECEP2=b
        MOV   ETAPE3,C   ;
        CPL   C
        MOV   ETAPE0,C   ; et ETAPE0=0
; activer étape 4 ?
        MOV   C,ETAPE3   ; si ETAPE4 et RECEP4 =1
        ANL   C,RECEP4   ; RECEP4=v
        MOV   ETAPE3,C   ;
        CPL   C
        MOV   ETAPE3,C   ;
; traitement de la convergence en OU
        MOV   C,ETAPE2   ; t1= ETAPE2 . x
        ANL   C,RECEP5   ; t1 est un bit de la zone
        MOV   t1,C       ; adressable au niveau du bit
        MOV   C,ETAPE4   ; C= ETAPE4 . w
        ANL   C,RECEP6   ;
        ORL   C,t1       ; C = ETAPE2.x + ETAPE4.w
        MOV   ETAPE0,C   ; si C=1 activer ETAPE0
        CPL   C
        MOV   ETAPE2,C   ; et désactiver ETAPE2
        MOV   ETAPE4,C   ; ainsi que ETAPE4
```

On désactive les étapes 2 et 4 bien qu'une seule ait été active. Le traitement proposé pour ce type de problème peut bien sûr être conçu de façon différente.

14

Aide au développement matériel et logiciel

Élaboration d'un projet

On posera ici comme à priori que le projet à réaliser, le sera à partir de la mise en œuvre d'un microcontrôleur en tant que commande du système.

Les différentes analyses réalisées à partir du cahier des charges, et formant les bases d'une bonne conception, ne seront pas développées dans ce chapitre.

Notons simplement que :

1 — Le cahier des charges du projet doit exprimer le plus clairement possible :

— les fonctions et les services attendus ;

— les performances souhaitées ;

— les limites tolérées ;

— les contraintes de l'environnement.

2 — Les outils d'analyses choisis doivent permettre de :

— définir la structure matérielle ;

— préparer l'organisation du logiciel ;

3 — Toute phase de conception doit s'accompagner d'une documentation bien souvent utile au moment des phase de test et de recherche d'erreurs

Choix de l'organisation matérielle

Définir la structure matérielle revient à :

— décrire l'interfaçage nécessaire entre la partie **commande** (le microcontrôleur) et la partie **opérative** du système ;

— préciser les relations avec l'environnement au système.

L'interfaçage avec la partie OPERATIVE peut s'exprimer en terme de :

— nombre d'entrées ;

— nature des signaux à traiter ;

— nombre de sorties.

Les relations avec l'environnement peut nécessiter :

— des entrées (clavier de commande, capteurs etc...)

— des sorties (afficheurs, voyants, etc...)

— un support normalisé d'échange d'informations (port série ou parallèle etc...)

Une fois l'analyse de conception amorcée tant au plan logiciel qu'au plan des besoins matériels il est possible d'effectuer des choix qui conduiront à la réalisation du support matériel de l'application.

Programme interne ou externe ?

Les éléments à prendre en compte pour donner une réponse à cette question peuvent se résumer de la façon suivante :

Si le projet doit être réalisé industriellement en production de masse la solution est alors toute trouvée ! Faites programmer vos microcontrôleurs lors de leur fabrication par le constructeur.

Pour une production unitaire ou de petite série, il reste l'alternative d'utiliser :

— soit, un microcontrôleur avec EPROM intégrée ;

— soit, un microcontrôleur sans mémoire programme, complété par une EPROM standard câblée en externe.

Microcontrôleur avec EPROM intégrée

Principaux avantages :

— gain d'encombrement sur le circuit imprimé ;

— tous les ports d'entée/sorties restent disponibles à condition que l'application ne nécessite pas de mémoire externe de données ;

— possibilité de protéger le programme contre les copies illicites.

Inconvénients :

— prix de revient souvent plus élevé que la combinaison microcontrôleur + EPROM ;

— impossibilité d'utiliser un simulateur d'EPROM lors des phases de tests de l'application ;

— nécessité de posséder un "programmateur d'EPROM" supportant l'adaptation au microcontrôleur utilisé.

L'utilisation d'un outil au coût plus élevé, comme l'émulateur, est donc nécessaire pour réaliser les mises au point de l'application.

Microcontrôleur + EPROM externe

Les avantages et inconvénients de cette solution découlent de l'analyse du cas précédent :

Avantages :

— prix de revient ;

— possibilité de moduler la taille de la mémoire programme ;

— un simulateur d'EPROM peut suffire lors des phases de tests de l'application ;

— utilisation d'un "programmateur d'EPROM" classique pour la mise en place du code ;

— mise à jour plus économique car l'échange standard se limite au remplacement de l'EPROM.

Inconvénients :

— protection du programme impossible ;

— mobilisation de deux ports en guise de bus externes ;

— augmentation relative de la complexité du câblage.

Les deux derniers inconvénients ne sont plus à prendre en compte si une mémoire externe de données s'avère nécessaire dans le cadre de l'application. Avant d'envisager une telle mémoire de données, il convient d'utiliser au mieux la mémoire interne.

Organisation de la RAM interne

Définir l'utilisation de la RAM interne revient à réserver des zones mémoires à des fonctions propres à l'application envisagée. L'affectation détaillée des 128 octets (ou 256 pour les versions 8052) reste donc une suite de cas particuliers. Il est possible malgré tout de suivre quelques règles d'utilisation.

Zone mémoire 0 à 1FH :

Cette zone correspond aux quatre banques de registres. La banque 0, active à l'initialisation, peut conserver son rôle dans le traitement principal du programme. Les trois autres banques peuvent être consacrées aux sous-programmes d'interruption. Il est alors possible de préserver l'état des 8 registres en cours d'utilisation, et d'activer 8 autres registres dédiés au traitement de l'interruption. Une banque inutilisée peut être considérée comme huit octets de mémoire dont l'utilisation sera à définir.

Zone mémoire 20H à 2FH :

Les seize octets de cette zone étant adressables au niveau du bit, elle devra être utilisée en priorité comme zone de variables binaires (indicateurs d'état de traitement, image d'une entrée ou d'une sortie logique etc...)

Zone mémoire 30H à 7FH :

Cette troisième partie de la mémoire peut être utilisée pour placer les variables de format 8 bits (ou plus) du programme. Elle peut aussi être utilisée comme tampon de réception et d'émission de la communication série.

Il faut se souvenir que le pointeur de pile est initialisé de la valeur 07. La pile d'origine se situe donc sur la banque 1 de registres. Il est souvent conseillé de modifier la valeur du pointeur SP et de placer la pile au dessus des autres variables de l'application. Si l'on utilise une version du microcontrôleur dotée de 256 octets, la pile peut empiéter sur la zone mémoire située au delà de 7FH. Rappelons d'ailleurs que cette zone n'est adressable qu'en mode indirect, ce qui convient parfaitement à la gestion de la pile. La figure 14.1 résume une utilisation classique de la RAM interne.

C'est après avoir attribué une fonction à chaque zone de la RAM interne, qu'il sera possible d'estimer la nécessité de compléter cette mémoire interne de données par un composant mémoire externe. L'éventualité d'étendre la mémoire de données en mode

externe s'accompagne de la réquisition du port P0, d'une partie (voir de l'ensemble) du port P2 et de l'adjonction d'un circuit de démultiplexage. Cette remarque est sans objet si la solution microcontrôleur + EPROM externe est retenue.

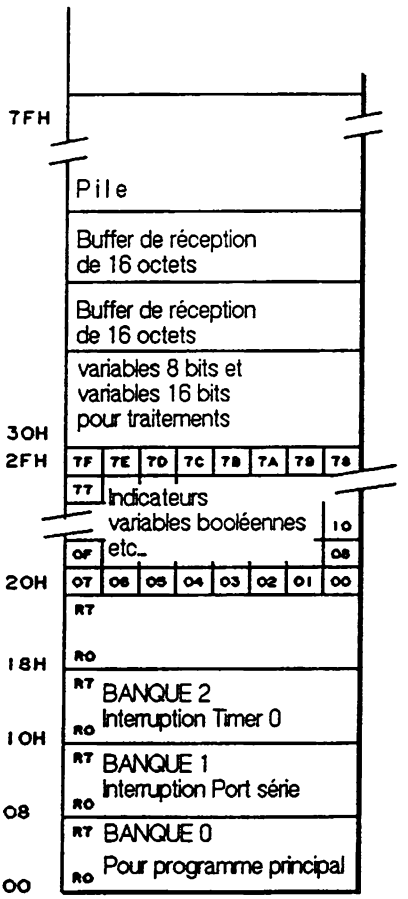


Figure 14.1 : Utilisation de la RAM interne

Composants de la famille MCS-80/85

La commercialisation des microprocesseurs 8 bits 8080 et 8085 par la société INTEL fut accompagnée par la mise sur le marché d'un bon nombre de composants périphériques qui répondaient au souci de créer autour d'une unité centrale un environnement minimum et ce avec un minimum de composants. La famille 8051 a

hérité d'un bon nombre de concepts déjà utilisés sur le microprocesseur 8085 :

— notion de bus multiplexe entre adresse et donnée
— commande de la périphérie par des signaux de même nature (ALE, $\overline{RD}$, $\overline{WR}$ etc.).

Il est donc possible de compléter les fonctions périphériques d'un microcontrôleur de la famille 8051 avec des composants de la famille MCS 80/85. La figure 14.2 propose, en deux circuits intégrés, une commande de système ne comportant pas moins de 41 lignes d'E/S (plus une entrée et une sortie pour communication série). Le circuit 8155 (ou sa version CMOS le 81C55) intègre en plus un timer de 14 bits et 256 octets de RAM qui viennent compléter les 256 octets internes au 87C52.

Extension de la mémoire programme à plus de 64 Ko

Un bon nombre de projets industriels sont assujettis à l'évolution de leur version initiale pour des versions plus sophistiquées. La création ou l'amélioration de fonctions, un dialogue homme/machine de plus en plus performant, la commande de nouveaux périphériques sont autant d'éléments qui entraînent rapidement un programme à dépasser la taille de 64 Ko. La solution serait alors d'abandonner le microcontrôleur 8 bits doté d'un bus d'adresses 16 bits, pour doter la nouvelle version d'un microcontrôleur 16 bits capable de gérer un espace d'adresses d'au moins 1 Mo. Un tel changement entraîne une révision complète du projet. Il faut reprendre toute la conception matérielle et logicielle du produit et en plus prévoir un nouvel équipement de développement !

Si certaines familles de microcontrôleurs se sont récemment enrichies de versions équipées d'une unité de gestion de la mémoire (MMU), ce n'est, hélas, pas encore le cas de la famille 8051.

Il est relativement simple, en envisageant le sacrifice de quatre lignes de port d'E/S en plus des ports P0 et P2 déjà mobilisés pour le bus d'adresses de 16 bits, d'augmenter le champ d'adresses du 8051.

Le schéma de la figure 14.3 propose une mise en œuvre de ce principe. Tant que le bit P2.7 reste à 0, les lignes d'adresses A15, A16, A17 et A18 de L'EPROM sont maintenues à 0. On exploite alors le code programme logé dans la PAGE 0 de 32 Ko (de 0 à 7FFFH). Selon la valeur des quatre bits du port P0 (P1.0 à P1.3), lorsque P2.7 est mis à 1, on accède à une des 16 pages de 32 Ko placées en superposition les unes par rapport aux autres (overlay). Vue du processeur, la mémoire programme est constituée d'un segment classique de 0 à 7FFFH et de 16 segments superposés de 8000H à FFFFH. Un seul de ces segments n'est accessible à la fois et donc un module de programme ne doit pas chevaucher deux pages en recouvrement, car un accès à un segment paginé doit être précédé d'une commutation de page et cette commutation de page doit être assurée par le logiciel.

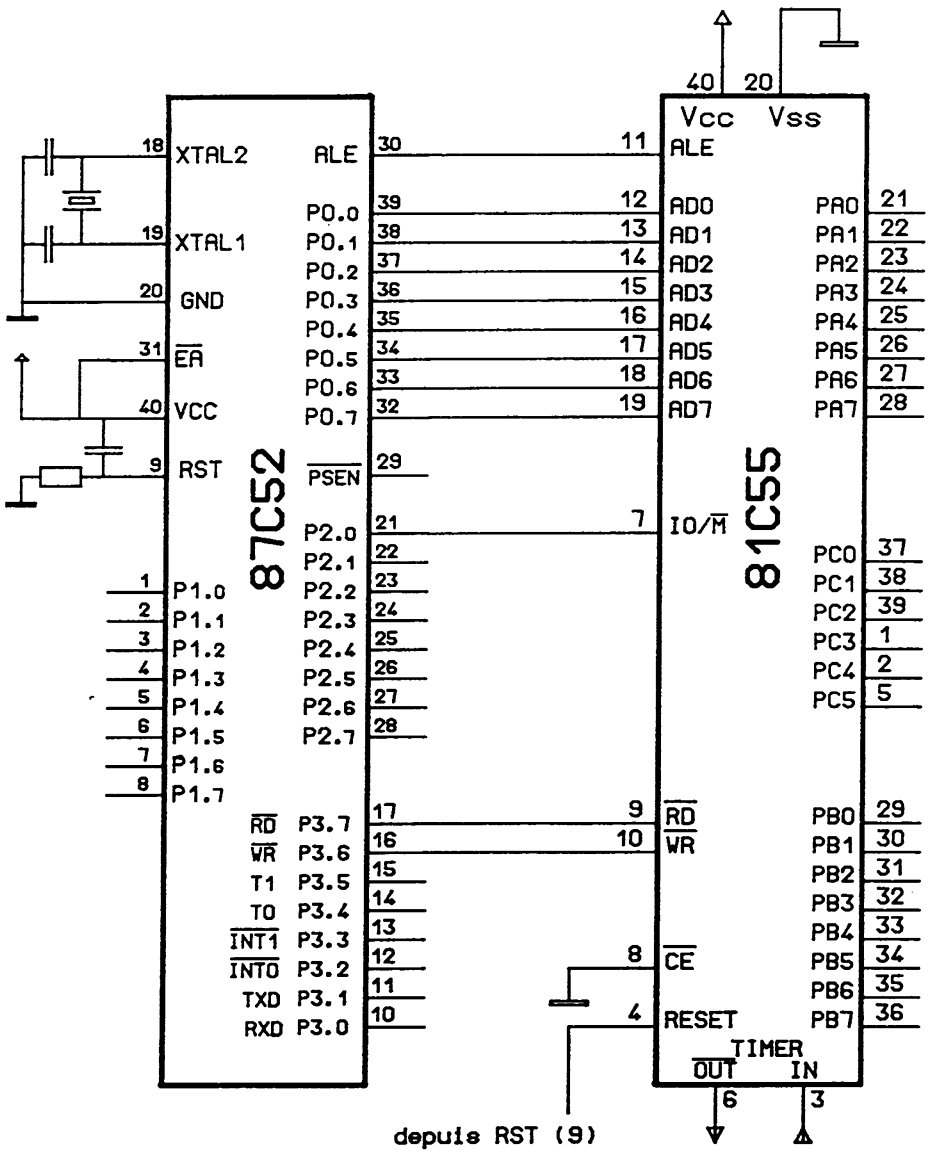


Figure 14.2 : Une commande de système en deux circuits intégrés

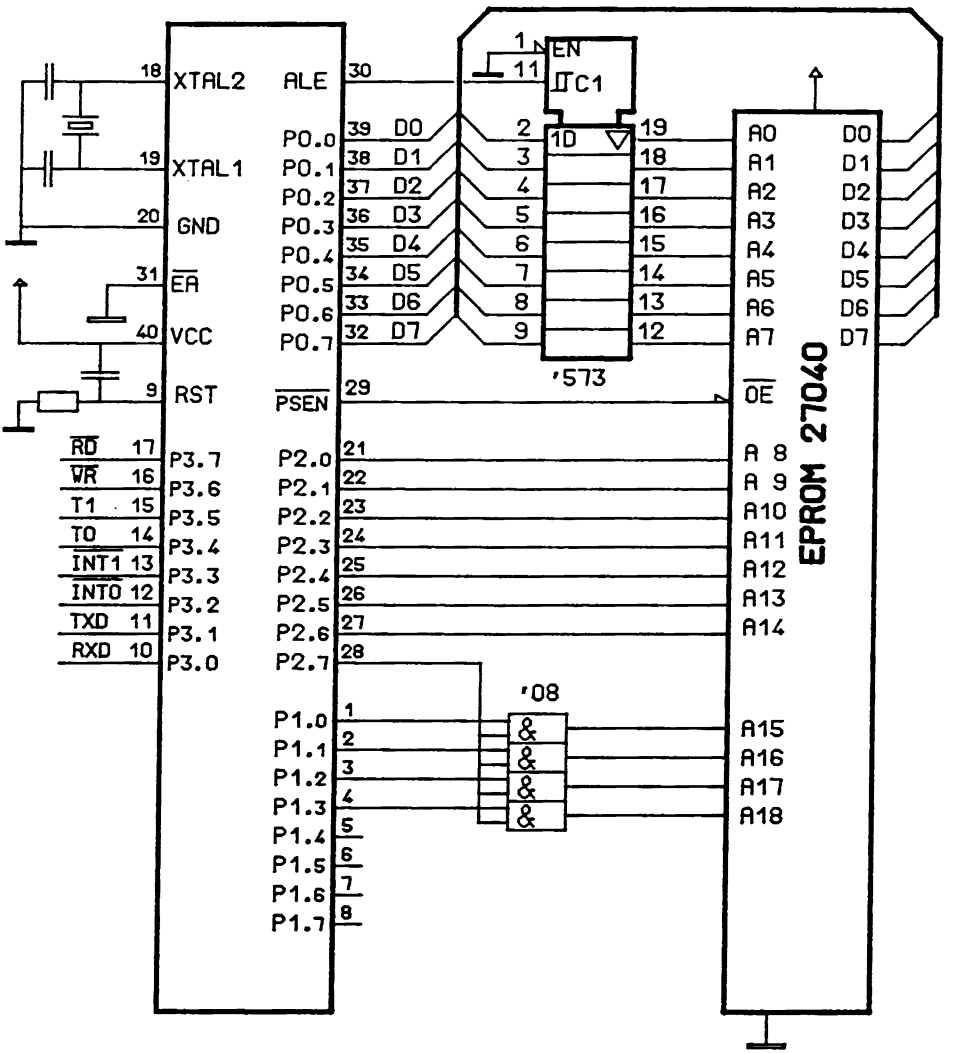


Figure 14.3 : Une mémoire programme de 512 Ko

Les assembleurs ou compilateurs prévus pour le 8051 ne peuvent pas, dans la majorité des cas, prendre à leur compte la gestion d'un programme supérieur à 64 Ko. Le programme devra donc être écrit en plusieurs segments et l'on confiera à un programme EDITEUR de LIENS spécialisé la gestion de la pagination du programme. La société RAISONANCE propose à cet effet un éditeur de liens étendu (LX-51) capable de gérer jusqu'à 31 pages de 32 Ko. Il est possible ainsi d'atteindre 1 Mo de mémoire programme.

Développement du logiciel

Après analyse et conception des traitements à effectuer, le développement d'un programme comporte trois phases :

— écriture du fichier source ;
— génération du code exécutable ;
— tests et mise au point.

Il y a quelques années, la réalisation de ces différentes opérations nécessitait des outils matériels et logiciels forts coûteux. En considérant la démocratisation des prix des programmes capables de produire le code objet exécutable, il est déconseillé d'établir "à la main" cette liste de valeurs, n'ayant de sens que pour le microcontrôleur ! Le risque d'erreurs est trop important et représente une dépense de temps et d'énergie peu justifiée. La plupart des programmes ASSEMBLEUR existent à l'heure actuelle dans des versions capables de fonctionner sur des micro-ordinateurs dits "compatibles PC". Chaque ASSEMBLEUR possède quelques particularités d'utilisation. Les exemples des chapitres précédents utilisaient des DIRECTIVES d'assemblage du programme ASM51 du constructeur INTEL. Les particularités d'un autre assembleur proposé par la société RAISONANCE seront évoquées dans les paragraphes suivants. Il faut signaler que cette société propose un ensemble EDITEUR/ASSEMBLEUR (EMA-51) à un prix "spécial enseignement" fort intéressant.

Fichier SOURCE

A l'aide d'un programme "traitement de texte" ou d'un EDITEUR, le programme est rédigé sous forme d'une suite d'instructions reconnues par le processeur (chapitre 7). Pour plus de lisibilité les instructions sont écrites sous leur forme mnémonique.

C'est le programme dit "assembleur" qui se chargera de remplacer cette représentation de l'instruction par son code binaire.

Afin de simplifier le travail de rédaction, des "pseudo-instructions" ou DIRECTIVES pour l'assembleur sont utilisables. Il est possible aussi de remplacer des valeurs numériques, peu évocatrices à leur lecture, par des symboles. Il est bien sûr conseillé de choisir des symboles ayant un sens pour améliorer la "lisibilité" du programme.

Toute cette démarche a pour but d'établir un document le plus clair possible afin de limiter les risques d'erreurs, surtout lorsque le programme commence à prendre de l'ampleur.

Il faut remarquer que certains ASSEMBLEUR ne reconnaissent pas naturellement les symboles des registres à fonction spéciale.

exemple : CLR 98H ; mise à 0 de l'indicateur RI

peut s'écrire

CLR RI

à condition de définir RI comme étant égal à 98H. On parlera alors de DECLARATIONS.

La suite de déclarations correspondant aux registres à fonction spéciale étant un problème commun à tous les développements sur un même type de microcontrôleur, les concepteurs de programmes "assembleur" ont prévu une directive autorisant l'inclusion d'une liste de déclarations :

$INCLUDE pour ASM-51 INTEL

INCLUDE pour EMA-51 RAISONANCE

Ainsi :

$NOMOD51

supprime la reconnaissance implicite des SFR du 8051 et

$INCLUDE (REG52.PDF)

permet de récupérer dans le fichier REG52.PDF les nouvelles déclarations propres au 8052.

Cette directive peut être utilisée pour lier au fichier source principal d'autres fichiers comportant des déclarations propres à l'application, ou même, des segments de programme. L'assembleur considère l'ensemble des fichiers inclus comme faisant partie du fichier principal. Il faut quand même se méfier du temps d'assemblage qui pourrait résulter d'un trop grand nombre de fichiers liés ainsi les uns aux autres !

Programmes de moyenne importance

Dans le cas d'un projet de moyenne importance le programme peut être développé sous la forme d'un seul fichier source. La directive INCLUDE peut être simplement utilisée pour la liste des registres à fonction spéciale. Dans ce cas de figure, les phases de développement se résument à :

— l'édition du fichier SOURCE ;

— l'assemblage de ce fichier.

Il résulte de cet assemblage, un fichier de codes exécutables qui peut très facilement être converti pour être placé dans l'EPROM du système. On peut considérer que cette procédure est satisfaisante pour des programmes dont le fichier source représente un document ne dépassant pas une trentaine de pages.

Développement de programmes plus conséquents

Si le développement s'avère plus complexe, il est possible d'envisager un découpage du programme en plusieurs fichiers. Les segments du programme doivent alors être "relogeables".

Une méthode consiste à développer, au sein d'un fichier principal, le corps du programme et de placer dans d'autres fichiers, des sous-programmes ou des modules utilisés par le corps principal. La directive "INCLUDE" n'est pas employée pour lier les différents fichiers. On lui préfère alors les directives PUBLIC et EXTERN pour que l'assembleur tolère l'absence apparente (dans le fichier principal) des sous-programmes et modules placés dans d'autres fichiers.

exemple :

On place dans un fichier baptisé BIBLIO un module assurant l'affichage sur un écran LCD. Ce module commence par un symbole (d'adresse relative) déclaré comme PUBLIC

. PUBLIC AFFICHE : ....

suite d'instructions

Par son attribut, le symbole AFFICHE devient alors "visible" par d'autres fichiers lors d'une édition de liens.

Par contre dans le fichier principal qui appelle par l'instruction "LCALL AFFICHE ", le module situé dans le fichier BIBLIO, le symbole AFFICHE aura été déclaré au préalable comme EXTERN. Dans ce cas, l'assembleur génère des suites de codes dites "relogeables" et l'utilisation d'un autre programme, l'ÉDITEUR de LIENS va se charger de gérer les références entre fichiers.

L'éditeur de liens a pour fonction de regrouper les fichiers de modules relogeables qui ont été assemblés séparément afin de produire finalement un fichier unique, directement utilisable pour la programmation d'une EPROM.

La démarche complète du développement se résume de la façon suivante :

— Éditions des fichiers sources ;

— Assemblage des différents modules ;

— Édition de liens pour générer le code absolu.

Pour plus de détails sur ces méthodes de travail, il est vivement conseillé de consulter la "documentation constructeur" livrée avec les programmes "assembleur" et "éditeur de liens".

A titre d'information les tableaux 14.5 et 14.6 présentent quelques directives d'assemblage utilisées dans les exemples de programmes présentés dans ce livre.

Outils de développement

Dans les documentations des outils de développement on utilise souvent les termes "système hôte" et "cible".

L'ordinateur sur lequel on développe le programme constitue le système hôte. La cible correspond au support matériel de l'application en cours de développement. La cible peut être le circuit imprimé définitif de l'application ou une carte sur laquelle les composants sont interconnectés par une technique de câblage nommée "Wrapping". Sur la cible on trouve donc un microcontrôleur accompagné ou non d'une mémoire programme (EPROM).

Lorsque le programme est écrit et que l'assemblage, accompagné ou non d'une édition de liens, a permis d'obtenir un fichier de codes exécutables, ces codes sont encore dans le système hôte. En reliant au système hôte un "programmateur d'EPROM", il est possible de transférer le fichier dans une EPROM. L'EPROM est ensuite placée sur la cible et on procède au premier test. L'essai n'est pas concluant : que doit-on mettre en cause, le programme, le câblage ou les deux ! Pour lever les doutes sur la partie "matériel" il est possible de prévoir un petit programme de tests. Mais il faudra, à chaque nouvel essai, effacer l'EPROM et la programmer à nouveau. La suite de ces opérations prend du temps et une EPROM n'est pas "reprogrammable" indéfiniment.

Simulateur de programmes

Pour lever les soupçons sur la partie "programme", l'idéal serait de pouvoir demander au microprocesseur du système hôte de simuler le déroulement du programme. Ce genre de programme existe sur le marché. Il permet de suivre sur l'écran de l'ordinateur le déroulement du programme, ses actions sur les registres, indicateurs et zones mémoire de données. Il est même possible de simuler des événements externes tels que communication sur le port série, mesure de tension pour les microcontrôleurs équipés de convertisseur analogique numérique, etc...

Il devient possible de corriger un grand nombre d'erreurs avant de passer à un e sur la cible.

En plus des services rendus, ce genre de programme possède des qualités didactiq indéniables. L'acquisition d'un tel outil est indispensable pour un bureau d'étude ou centre de formation.

Simulateur d'EPROM

Le simulateur se substitue à l'EPROM placée sur la carte cible. C'est en fait mémoire RAM téléchargeable depuis le système hôte. L'avantage de ce simulateur de pouvoir tester et modifier rapidement un programme en supprimant to manipulation de composant. Son faible coût et le gain de temps qu'il apporte en f un outil indispensable. Une fois l'application au point, le programme sera placé d une véritable EPROM et prendra place définitivement sur la carte cible.

Émulateurs

Un émulateur remplace le microcontrôleur sur la carte cible. Sa conception, beauc plus complexe qu'un simulateur d'EPROM, en fait un produit généralement beauc plus cher. L'émulateur exécute les instructions, produit les signaux comme s modèle et ce dans un temps pratiquement identique. Mais en plus il est capable mémoriser des "traces" des événements et peut aussi fonctionner en mode pas à p Cet outil permet aussi bien une mise au point du logiciel qu'un dépistage efficace c problèmes dus au matériel. Son utilisation est recommandée dans les applicatic délicates où la mise au point du programme est très liée au support matériel du pro

ASM-51		FONCTION
CSEG	AT adr	définit le début d'un segment de code exécutable à partir de l'adresse absolue adr
DSEG	AT adr	définit le début absolu d'une zone RAM interne
XSEG	AT adr	définit le début absolu d'une zone RAM externe
BSEG	AT adr	définit le début d'une zone adressable au niveau du bit.
EQU		affecte une valeur numérique à un symbole
DB		Insère une suite de valeurs au format "octet"
DS	n	réserve n octets
DW		insère une suite de valeurs de format 16 bits.
DBIT	n	réserve et déplace compteur de bit adressable.
END		fin du programme.

Exemple de mise en œuvre :

```
NBR_FOIS        EQU 5           ;le symbole correspond à la valeur 5.
        DSEG    AT 30H          ;Dans la RAM interne à partir de l'adresse
MESURE :        DS  1           ;30H on réserve 1 octet pour la variable
                                ;MESURE puis à la suite (31 et 32H)
CALCUL :        DS  2           ;on réserve 2 octets pour CALCUL.

        BSEG    AT  0           ;A partir de l'adresse 20H le premier bit
FLAG_IT:        DBIT 1          ;d'adresse 0 est réservée à FLAG_IT
BASCULE :       DBIT 1          ;puis le bit suivant est réservé à BASCULE

        XSEG    AT  0           ;Dans la RAM externe à partir de 0
TAMPON :        DS  32          ;on réserve 32 octets pour TAMPON
CONFIG :        DS  5           ;puis à la suite 5 octets pour CONFIG

        CSEG    AT  0           ;segment de code absolu à l'adresse 0
        LJMP    DEPART          ;saut vers adresse DEPART après un RESET
                                ;cette instruction occupe 3 octets.
                                ;pas de code placé avant l'adresse 23H
        CSEG    AT  23H
SERIE :                         ;sous-programme d'interruption
        suite d'instructions
        ..............
        ..............
DEPART :                        ;début du traitement principal
        ..............
        ..............
        END                     ;fin du fichier
```

Tableau 14.5 : Quelques directives pour ASSEMBLEUR ASM51

EMA-51		FONCTION
CODE		spécifie un segment de codes relogeables
CODE	AT adr	définit le début absolu d'un segment de codes
DATA		spécifie un segment en RAM interne relogeable
DATA	AT adr	définit le début absolu d'un segment de RAM interne
XDATA		spécifie un segment de données en RAM externe
XDATA	AT adr	définit le début absolu d'une zone RAM externe
BIT		spécifie un segment adressable au niveau du bit
BIT	AT adr	définit le début absolu d'une zone "BIT"
NUMBER		affecte une valeur numérique à un symbole
DB	n	réserve n octets
DB	=	Insère une suite de valeurs au format "octet"

Exemple :

```
NB_FOIS         NUMBER 5        ;le symbole correspond à la valeur 5.

        DATA    AT 30H          ;Dans la RAM interne à partir de l'adresse
MESURE:         DB  1           ;30H on réserve 1 octet pour la variable
                                ;MESURE puis à la suite (31 et 32H)
CALCUL:         DB  2           ;on réserve 2 octets pour CALCUL.

        BIT     AT  0           ;A partir de l'adresse 20H le premier bit

FLAG_IT :       DB  1           ;d'adresse 0 est réservée à FLAG IT
BASCULE:        DB  1           ;puis le bit suivant est réservé à BASCULE

        XDATA   AT  0           ;Dans la RAM externe à partir de 0
TAMPON:         DB  32          ;on réserve 32 octets pour TAMPON
CONFIG:         DB  5           ;puis à la suite 5 octets pour CONFIG
        CODE    AT  0           ;segment de code absolu à l'adresse 0
        LJMP    DEPART          ;saut vers adresse DEPART après un RESET
                                ;cette instruction occupe 3 octets.
                                ;pas de code placé avant l'adresse 23H
        CODE    AT  23H
SERIE:                          ;sous-programme d'interruption
        suite d'instructions
        ..............
        ..............
DEPART:                         ;début du traitement principal
        ..............
        ..............
DB      = "SALUT"               ;insère les codes ASCII de la chaîne SALUT
```

Tableau 14.6 : Quelques directives pour ASSEMBLEUR EMA-51

15

Ressources logicielles

Module arithmétique

Les sous-programmes de ce paragraphe, inspirés d'une note l'application de Rick Schue publiée par Intel Corporation, permettent de réaliser des opérations arithmétiques sur des variables d'au moins 16 bits. Ils utilisent tous une même zone mémoire de 17 octets qui doit être située en RAM interne adressable aussi bien en mode direct qu'en mode indirect par registre. Dans l'exemple de déclaration de cette zone mémoire, l'adresse 30H a été retenue. Les variables REG_A et REG_B représentent deux registres "opérandes", et ACCU, le registre de 32 bits dans lequel le résultat de l'opération est rangé. Le registre REMAIN représente le reste d'une division, il doit être de 24 bits pour satisfaire les besoins de l'algorithme de la division 32 bits par 16 bits. Le registre BCD reçoit les trois octets d'une conversion en code BCD d'un entier de 16 bits. La conversion de la valeur FFFFH place dans ce registre les valeurs suivantes :

BCD+2	BCD+1	BCD
06	55	35

Avant d'appeler une de ces fonctions de calcul, il suffit de pointer avec R0 et R1 les deux variables à traiter. Le sous-programme se charge alors de placer dans les registres REG_A et REG_B les valeurs des deux variables, puis effectue le calcul, et replace le pointeur R0 sur le résultat (poids faible de ACCU).

Un petit programme est donné comme exemple d'utilisation de ces sous-programmes :

17 octets réservés a partir de 30

```
                                    DSEGAT (30H)
;
DIVCTR :   DS      1        ; compteur pour division
REG_A :    DS      4        ; registre A 16 bits ou 32 bits pour div. 32/16
REMAIN :   DS      3        ; reste de la division (extension de A)
REG_B :    DS      2        ; registre B 16 bits
ACCU :     DS      4        ; accumulateur 32 bits
BCD :      DS      3        ; registre résultat de la conversion BCD
```

Exemple de programme utilisant le module arithmétique
Le résultat d'une conversion analogique numérique 0/5 volts sur 10 bits
représente une tension entre 0 et 400 volts

```
;variables spécifiques à l'exemple
MESURE :   DS      2        ; résultat mesure faite par un C.A.N
CALIBRE :  DS      2        ;
ECHELLE :  DS      2
;POINT D'ENTREE APRES RESET
           CSEG    AT 0000H ; ADRESSE du vecteur RESET
           ORG     0H
           JMP     DEBUT
ORG        100H
DEBUT :    MOV     DPTR,#400    ; calibre 400 volts par exemple
           MOV     CALIBRE,DPL  ; mis en place en mémoire
           MOV     CALIBRE+1,DPH
           MOV     DPTR,#1024   ; Echelle de 1024 pour 10 bits
           MOV     ECHELLE,DPL
           MOV     ECHELLE+1,DPH
;Après cette initialisation faire le calcul
           MOV     R1,#CALIBRE  ; R1 pointe le calibre
           MOV     R0,#MESURE   ; R0 pointe la mesure
           LCALL   MUL_AB       ; faire MESURE x CALIBRE
           MOV     R1,#ECHELLE  ; R1 pointe la valeur de l'échelle
```

```
        LCALL   DIV_TS    ; Faire (MESURExCALIBRE)/ECHELLE
        LCALL   HEX_BCD   ; convertir en code BCD
```

Les valeurs de BCD+1 et de BCD peuvent alors être affichées
BCD représente dizaine et unité du résultat et les quatre bits de poids faible de BCD+1 représentent le chiffre des centaines de volts.

Chargement dans REG_A d'un 16 bits pointé par R0

```
LOAD_16:  MOV   REG_A+0,@R0
          INC   R0
          MOV   REG_A+1,@R0
          MOV   REG_A+2,#0
          MOV   REG_A+3,#0
          RET
```

Chargement dans REG_A d'un 32 bits pointé par R0

```
LOAD_32:  MOV   REG_A+0,@R0
          INC   R0
          MOV   REG_A+1,@R0
          INC   R0
          MOV   REG_A+2,@R0
          INC   R0
          MOV   REG_A+3,@R0
          RET
```

Multiplication 16 bits par 16 bits

L'algorithme utilisé est calqué sur la méthode que l'on utilise lorsqu'on effectue une multiplication "à la main". En admettant que A, B, C et D représentent des octets, AB et CD représentent des valeurs de 16 bits :

soit l'opération AB x CD

On effectue D x B puis D x A, ensuite C x B et C x A.

Il est très facile de multiplier deux octets entre eux grâce à l'instruction MUL AB. Seule la gestion des résultats et des sommes avec décalage demande un peu d'attention.

MULTIPLICATION 16 X 16 bits

entrée : R0 et R1 pointe le poids faible des deux variables

sortie : résultat dans ACCU pointé par R0

```
MUL_AB :    ACALL   LOAD_16         ;charger REG_A avec variable (R0)
            MOV     REG_B,@R1       ;placer variable pointée par R1
            INC     R1              ;dans REG_B
            MOV     REG_B+1,@R1
            ACALL   AB_MUL          ; effectuer multiplication
            MOV     R0,#ACCU        ; poids faible résultat pointé par R0
            RET

AB_MUL :    MOV     ACCU+2,#0       ; poids fort accumulateur à 0
            MOV     ACCU+3,#0       ;
            MOV     A,REG_A         ; multiplier les poids faibles
            MOV     B,REG_B         ; ( idem D x B)
            MUL     AB
            MOV     ACCU,A          ; résultat provisoire dans ACCU
            MOV     ACCU+1,B
            MOV     A,REG_A+1       ; poids fort A par poids faible B
            MOV     B,REG_B         ; (idem D x A)
            MUL     AB
            ADD     A,ACCU+1        ; somme avec partiel précédent
            MOV     ACCU+1,A        ; avec un décalage vers gauche
            MOV     A,B
            ADDC    A,ACCU+2        ; prise en compte d'un report
            MOV     ACCU+2,A        ; vers ACCU+2
            JNC     AB_M1           ;
            INC     ACCU+3          ; et vers ACCU+3
AB_M1 :     MOV     A,REG_A         ; multiplier poids faible A par
            MOV     B,REG_B+1       ; poids fort de B
            MUL     AB              ; (idem C x B)
            ADD     A,ACCU+1
            MOV     ACCU+1,A
            MOV     A,B
            ADDC    A,ACCU+2
```

```
            MOV     ACCU+2,A
            JNC     AB_M2
            INC     ACCU+3
AB_M2 :     MOV     A,REG_A+1       ; multiplier poids fort de A
            MOV     B,REG_B+1       ; avec poids fort de B
            MUL     AB              ; ( idem A x B)
            ADD     A,ACCU+2        ; somme avec décalage
            MOV     ACCU+2,A
            MOV     A,B
            ADDC    A,ACCU+3        ; en tenant compte des reports
            MOV     ACCU+3,A
            RET
```

Division 32 bits par 16 bits

Le résultat de cette division est situé dans ACCU sur 32 bits. L'exécution de ce sous-programme nécessite entre 1700 et 1960 cycles machine. Le temps d'exécution pourrait être considérablement réduit en admettant que le quotient soit limité à 16 bits, mais il faut alors se méfier des risques de débordement.

---

diviser (R0) (32 bits) par (R1) (16bits)
sortie :
quotient dans ACCU (32bits) pointé par R0
et (R1) reste sur 16 bits

```
DIV_TS :    ACALL   LOAD_32         ; (R0) placé dans REG_A
            MOV     REG_B,@R1
            INC     R1
            MOV     REG_B+1,@R1
            ACALL   DIV_32
            MOV     R0,#ACCU        ; quotient pointé par R0
            MOV     R1,#REMAIN      ; reste pointé par R1
            RET
;
DIV_32 :    MOV     REMAIN,#0       ;reste à 0
            MOV     REMAIN+1,#0
            MOV     REMAIN+2,#0
```

```
                MOV     DIVCTR,#32      ; reprendre DIV_T1 32 fois
;
DIV_T1 :        ACALL   SH_DIVI         ; décalage gauche du dividende
                CLR     C               ; tester si reste > diviseur
                MOV     A,REMAIN
                SUBB    A,REG_B
                MOV     A,REMAIN+1
                SUBB    A,REG_B+1
                MOV     A,REMAIN+2
                SUBB    A,#0
                JNC     SUB_T           ;reste > diviseur alors soustraire
                SJMP    QUOT_T          ;sinon introduire un 0 dans résultat
SUB_T:          CLR     C
                MOV     A,REMAIN        ;soustraire diviseur du reste partiel
                SUBB    A,REG_B
                MOV     REMAIN,A
                MOV     A,REMAIN+1
                SUBB    A,REG_B+1
                MOV     REMAIN+1,A      ;
                MOV     A,REMAIN+2
                SUBB    A,#0
                MOV     REMAIN+2,A      ;
;
QUOT_T :        CPL     C               ; introduire report dans
                MOV     A,ACCU          ; quotient
                RLC     A               ; 0 si reste < diviseur
                MOV     ACCU,A          ; 1 si reste > diviseur
                MOV     A,ACCU+1
                RLC     A
                MOV     ACCU+1,A
                MOV     A,ACCU+2
                RLC     A
                MOV     ACCU+2,A
                MOV     A,ACCU+3
                RLC     A
                MOV     ACCU+3,A
                DJNZ    DIVCTR,DIV_T1   ;
                RET
```

```
SH_DIVI :   CLR C
            MOV   A,REG_A
            RLC   A
            MOV   REG_A,A
            MOV   A,REG_A+1
            RLC   A
            MOV   REG_A+1,A
            MOV   A,REG_A+2
            RLC   A
            MOV   REG_A+2,A
            MOV   A,REG_A+3
            RLC   A
            MOV   REG_A+3,A      ; bit poids fort dans Cy
            MOV   A,REMAIN       ; placer report dans poids faible
            RLC   A              ; du reste partiel
            MOV   REMAIN,A
            MOV   A,REMAIN+1     ; sur 16 bits
            RLC   A
            MOV   REMAIN+1,A
            MOV   A,REMAIN+2
            RLC   A
            MOV   REMAIN+2,A
            RET
```

Division 16 bits par 16 bits

DIVISION 16/16
diviser (R0) (16 bits) par (R1) (16bits)
sortie :
quotient dans ACCU (16 bits) pointé par R0
et (R1) reste sur 16 bits

```
DIV_SS :    ACALL  LOAD_16       ; (R0) placé dans REG_A
            MOV    REG_B,@R1
            INC    R1
            MOV    REG_B+1,@R1
            ACALL  DIV_32
            MOV    R0,#ACCU      ; quotient pointé par R0
```

```
                MOV     R1,#REMAIN      ; reste pointé par R1
                RET
DIV_16:         MOV     REMAIN,#0       ; reste à 0
                MOV     REMAIN+1,#0
                MOV     REMAIN+2,#0
                MOV     DIVCTR,#16      ; compteur pour opération
DIV_S1:         ACALL   SH_DIVI_S       ; décalage gauche du dividende
                CLR     C               ; tester si reste > diviseur
                MOV     A,REMAIN
                SUBB    A,REG_B
                MOV     A,REMAIN+1
                SUBB    A,REG_B+1
                MOV     A,REMAIN+2
                SUBB    A,#0
                JNC     SUB_S           ; reste > diviseur alors soustraire
                SJMP    QUOT_S          ; sinon introduire 0 dans résultat
SUB_S:          CLR     C
                MOV     A,REMAIN        ; soustraire diviseur du reste partiel
                SUBB    A,REG_B
                MOV     REMAIN,A
                MOV     A,REMAIN+1
                SUBB    A,REG_B+1
                MOV     REMAIN+1,A
                MOV     A,REMAIN+2
                SUBB    A,#0
                MOV     REMAIN+2,A      ;
QUOT_S:         CPL     C
                MOV     A,ACCU          ; Cy introduit dans quotient
                RLC     A
                MOV     ACCU,A
                MOV     A,ACCU+1
                RLC     A
                MOV     ACCU+1,A
                DJNZ    DIVCTR,DIV_S1
                RET
SH_DIVI_S:
                CLR     C
                MOV     A,REG_A
                RLC     A
                MOV     REG_A,A
```

```
        MOV     A,REG_A+1
        RLC     A
        MOV     REG_A+1,A
        MOV     A,REMAIN        ;placer report dans poids faible
        RLC     A               ; du reste partiel
        MOV     REMAIN,A
        MOV     A,REMAIN+1      ; sur 16 bits
        RLC     A
        MOV     REMAIN+1,A
        MOV     A,REMAIN+2
        RLC     A
        MOV     REMAIN+2,A
        RET
```

Conversion en format BCD

---

convertir en BCD le 16 bits pointé par R0
R0 pointe l'octet de poids faible des 16 bits
la conversion est rangée dans BCD+2 (poids fort)
BCD+1
BCD+0 (poids faible)

```
HEX_BCD :
        ACALL   LOAD_16
        MOV     REG_B,#10H
        MOV     REG_B+1,#27H    ; division par 10000
        ACALL   DIV_16
        MOV     BCD+2,ACCU      ; poids fort dans BCD+2
        MOV     R0,#REMAIN      ; reste pointé par R0
        ACALL   LOAD_16         ; charger reste dans REG_A
        MOV     REG_B,#0E8H     ; division par 1000
        MOV     REG_B+1,#3
        ACALL   DIV_16
        MOV     A,ACCU
        SWAP    A
        MOV     BCD+1,A
```

```
        MOV     R0,#REMAIN      ; reste pointé par R1
        ACALL   LOAD_16         ; charger reste dans REG_A
        MOV     REG_B,#64H      ; division par 100
        MOV     REG_B+1,#0
        ACALL   DIV_16
        MOV     A,ACCU
        ORL     BCD+1,A
        MOV     A,REMAIN        ; reste inférieur à 100
        MOV     B,#10           ; division par 10
        DIV     AB
        SWAP    A               ; A = dizaine
        ORL     A,B             ; placer unite
        MOV     BCD,A           ; rangement dizaine et unité
        RET
```

Division par 256

DIVISION DE L'ACCUMULATEUR de 32 bits par 256
Reste dans REMAIN (1 octet)

```
DIV_256 :   MOV     REMAIN,ACCU
            MOV     ACCU,ACCU+1
            MOV     ACCU+1,ACCU+2
            MOV     ACCU+2,ACCU+3
            MOV     ACCU+3,#0
            RET                     ; 12 cycles machine
```

Gestion d'un afficheur L.C.D

Le module de gestion proposé permet de commander un afficheur à cristaux liquides de deux lignes de 16 caractères. L'analyse de nombreuses documentations laisse apparaître une grande similitude de conception des différents modèles proposés par divers constructeurs. Seules la disposition des connections et la tension de polarisation nécessaire au réglage de contraste peuvent changer d'une marque à l'autre.

Les sous-programmes proposés permettent aussi de commander d'autres modèles d'afficheurs comportant un nombre de lignes (ou un nombre de caractères par ligne) différent du modèle pris comme exemple.

Chaque caractère possède une adresse. Les adresses des caractères de la deuxième ligne ne sont pas consécutives aux adresses des caractères de la première ligne. Ainsi, quelle que soit la longueur de ligne du modèle choisi (16, 20, 24 ou 40 caractères), l'adresse du premier caractère de la deuxième ligne est toujours 40H. Ces adresses permettent de positionner directement le curseur sur une zone déterminée de l'afficheur. Le code de contrôle à envoyer est composé de l'adresse (sur 7 bits) précédé d'un bit à 1. Dans le programme ci-après l'adresse 40H se traduit donc par le code de contrôle C0H.

Le schéma de la figure 15.1 propose une solution d'interfaçage entre le microcontrôleur et l'afficheur. Le décodeur d'adresses détermine des "pages" de 8 Ko dans l'espace de mémoire de données. La première page (de 0 à 1FFFH) est laissée libre pour une éventuelle mémoire RAM du type 6264. L'afficheur bloque les 8 Ko de la page suivante (de 2000H à 3FFFH), alors qu'un espace de quatre adresses suffit à sa commande. Cette solution de décodage peut être modifiée et améliorée mais il est important de maintenir les lignes A0 et A1 respectivement sur les entrées R/W et RS de l'afficheur.

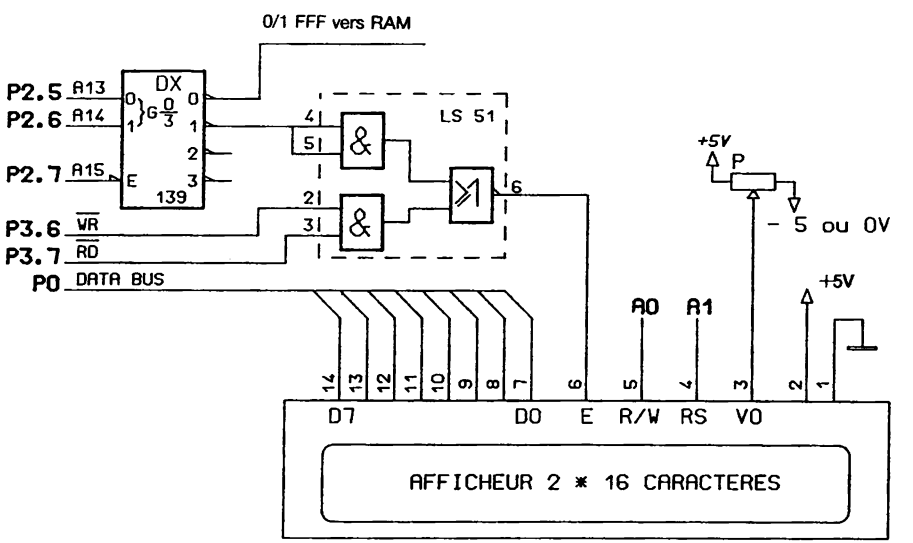


Figure 15.1 : Schéma de câblage de l'afficheur

Le tableau 15.2 précise le rôle de ces entrées de contrôle.

RS	R/W	fonction
A1	A0	
0	0	écriture dans le registre de contrôle
0	1	lecture du registre d'état (indicateur BUSY)
1	0	écriture d'un caractère (registre données)
1	1	lecture du registre de données

Note sur le sous-programme INI_AFF

La procédure d'initialisation de l'afficheur n'est garantie que si la mise sous tension de l'afficheur s'effectue dans le respect de la condition suivante : la tension entre les bornes 1 et 2 doit atteindre 5 volts dans un laps de temps compris entre 0.1 ms et 10 ms maximum. Dans le cas contraire l'initialisation du microcontrôleur interne à l'afficheur risque de ne pas être assurée correctement. Une procédure logicielle particulière doit alors forcer cette initialisation. Pour plus ample détail, il est conseillé de consulter la documentation constructeur du modèle d'afficheur utilisé.

GESTION AFFICHEUR

```
;
;NUL          EQU0          ; CODE ASCII 0
;ADRESSES RESULTANT DU CABLAGE DE L'AFFICHEUR
CTR_AFF      EQU 2000H     ; REGISTRE DE CONTROLE
STAT_AFF     EQU 2001H     ; REGISTRE D'ETAT
DATA_AFF     EQU 2002H     ; REGISTRE ECRITURE DE DONNEES
RD_DATA      EQU 2003H     ; REGISTRE LECTURE DE DONNEES
;
;PROGRAMME EXEMPLE
;
MAIN :       MOV     DPTR,#M_TEST    ;pointer le message
             LCALL   MESSAGE         ;et l'afficher
             MOV     A,#0C0H         ;placer le curseur en début
             LCALL   OUT_CTR         ;de la ligne 2
STOP :       SJMP    STOP            ;ARRET
```

ENVOIE VERS AFFICHEUR LA CHAINE en ROM
pointée PAR DPTR
FIN DE CHAINE SIGNALEE PAR "NUL"

```
MESSAGE :   CLR     A               ; récupérer donnée
            MOVC    A,@A+DPTR       ; pointée par DPTR
MES1 :      CJNE    A,#NUL,MES      ; terminaison ?
            RET                     ; oui alors fin
MES :       ACALL   OUT_CHAR        ; sinon envoi
            INC     DPTR            ; et refaire
            SJMP    MESSAGE
```

AFFICHEUR CURSEUR CLIGNOTANT

```
CURS_ON :   PUSH    ACC
            MOV     A,#0DH          ;curs on + BLINK
            ACALL   OUT_CTR
            POP     ACC
            RET
```

SUPPRIME CURSEUR

```
CURS_OFF :  PUSH    ACC
            MOV     A,#0CH
            ACALL   OUT_CTR
            POP     ACC
            RET
```

ENVOI A=CODE DE CONTROLE VERS AFFICHEUR

```
OUT_CTR :   PUSH    DPH
            PUSH    DPL
            ACALL   FREE_AFF
            MOV     DPTR,#CTR_AFF
            MOVX    @DPTR,A
            POP     DPL
            POP     DPH
            RET
```

ENVOI A = CARACTERE VERS AFFICHEUR

```
OUT_CHAR :  PUSH    DPH
            PUSH    DPL
            ACALL   FREE_AFF
            MOV     DPTR,#DATA_AFF
            MOVX    @DPTR,A
            POP     DPL
            POP     DPH
            RET
```

ATTENTE AFFICHEUR LIBRE
BIT 7 ← BUSY = 1 afficheur non disponible

```
FREE_AFF :  PUSH    ACC             ;sauver contexte
            PUSH    DPH
            PUSH    DPL
            MOV     DPTR,#STAT_AFF  ;pointer registre ETAT
FREE_0 :    MOVX    A,@DPTR         ;lire l'état
            JB      ACC.7,FREE_0    ;si BUSY=1 attendre
            POP     DPL             ;sinon afficheur
            POP     DPH             ;prêt à recevoir
            POP     ACC
            RET
```

INITIALISE L'AFFICHEUR

```
INI_AFF :   MOV     A,#38H          ;ACTIVE LES DEUX LIGNES
            ACALL   OUT_CTR
CLR_AF :    MOV     A,#01H          ;"HOME CURSOR"
            ACALL   OUT_CTR
            MOV     A,#0CH          ;display ON et curseur OFF
            ACALL   OUT_CTR
            MOV     A,#06           ;GAUCHE VERS DROITE
            ACALL   OUT_CTR
            RET
M_TEST :    DB      'AFFICHEUR OK !',0
```

Annexe

Précis de programmation

Mot d'état programme :

PSW adresse directe D0H adressable au niveau bit

	D7	D6	D5	D4	D3	D2	D1	D0
D0H	CY	AC	F0	RS1	RS0	OV	—	P

CY : indicateur de retenue (désigné aussi par la lettre C)
AC : indicateur de retenue auxiliaire indiquant un report du bit 3 vers le bit 4 (utilisé pour les opérations en code BCD).
F0 : indicateur F0. Sans fonction, mis à disposition de l'utilisateur.
RS1 : Sélection d'une banque de registres.
RS0 : Sélection d'une banque de registres.
OV : indicateur de débordement.
P : indicateur de parité.

RS1	RS0	banque active	adresse
0	0	0	00-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

Instructions modifiant les indicateurs

Instruction	Cy	Ov	AC	Instruction	Cy	Ov	AC
ADD	X	X	X	SETB C	1		
ADDC	X	X	X	CLR C	0		
SUBB	X	X	X	CPL C	X		
MUL	0	X		ANL C,BIT	X		
DIV	0	X		ANL C,/BIT	X		
DA	X			ORL C,BIT	X		
RRC	X			ORL C,/BIT	X		
RLC	X			MOV C,BIT	X		
CJNE	X						

TMOD : Mode de fonctionnement des TIMERS 0 et 1

adresse directe 89H

GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE : GATE = 1, le TIMER x est validé seulement lorsque sa broche d'entrée "INTx" est à l'état 1 et que le bit "TRx" du registre TCON est à 1.
GATE = 0, le TIMER est validé si "TRx" = 1.

C/T : Permet la sélection de la fonction TIMER :
C/T = 0 le TIMER est en fonction TEMPORISATEUR,
C/T = 1 le TIMER est en fonction COMPTEUR à partir des événements présents sur la broche d'entrée "Tx".

M1	M0	Mode
0	0	compteur 13 bits compatible au microcontrôleur MCS-48
0	1	compteur 16 bits
1	0	compteur 8 bits à rechargement automatique. Le contenu de THx est rechargé dans TLx lorsque celui-ci repasse à 0.
1	1	TIMER 0: Le registre TL0 devient un compteur 8 bits contrôlé normalement par les bits de contrôle du TIMER 0. Par contre, TH0 est configuré en temporisateur 8 bits contrôlé par les bits de contrôle du TIMER 1.
1	1	TIMER 1: compteur 1 à l'arrêt.

TCON : Contrôle des Timers 0 et 1

adresse directe 88H adressable au niveau bit

8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 : Indicateur de débordement du TIMER 1. Mis à 1 automatiquement lorsque le compteur repasse à 0. Si l'interruption correspondante est autorisée, cet indicateur est remis à 0 automatiquement lorsque le sous-programme d'interruption est exécuté.

TR1 : Bit de déclenchement du TIMER 1. Il doit être positionné par logiciel à 1 (pour lancer) ou 0 (pour arrêter) le TIMER.

TF0 : Indicateur de débordement du TIMER 0. Identique à TF1.

TR0 : Bit de déclenchement du TIMER 0. même fonctionnement que TR1.

IE1 : Indicateur de transition sur la source externe INT1. Cet indicateur est mis à 1 automatiquement lorsqu'une transition est détectée sur la borne INT1. Lors du service de l'interruption cet indicateur est remis à 0.

IT1 : Sélection du type d'événement devant déclencher l'interruption depuis le source INT1.
IT1 = 0 interruption sur détection d'un niveau 0 sur INT1.
IT1 = interruption sur transition négative (1 vers 0).

IE0 : Indicateur de transition sur la source externe INT0.
Fonctionnement identique à IE1.

IT0 : IT0 = interruption sur détection d'un niveau 0 sur INT0.
IT0 = interruption sur détection d'une transition négative.

Initialisation d'un Timer

Les valeurs données dans les quatre tableaux suivants permettent d'initialiser un seul Timer à la fois. Pour initialiser les deux timers simultanément, la valeur d'initialisation s'obtient par somme logique des deux valeurs.

Timer 0 comme temporisation

MODE	Fonction	VALEUR DE TMOD	
		contrôle interne	contrôle externe
0	TIMER 13 bits	00H	08H
1	TIMER 16 bits	01H	09H
2	RECHAR.AUTO 8 bits	02H	0AH
3	2 timers 8 bits	03H	0BH

Timer 0 comme compteur

MODE	Fonction	VALEUR DE TMOD	
		contrôle interne	contrôle externe
0	TIMER 13 bits	04H	0CH
1	TIMER 16 bits	05H	0DH
2	RECHAR.AUTO 8 bits	06H	0EH
3	2 timers 8 bits	07H	0FH

Timer 1 comme temporisation

MODE	Fonction	VALEUR DE TMOD	
		contrôle interne	contrôle externe
0	TIMER 13 bits	00H	80H
1	TIMER 16 bits	10H	90H
2	RECHAR.AUTO 8 bits	20H	A0H
3	ne fonctionne pas	30H	B0H

Note :
contrôle interne : le timer est lancé ou arrêté par le bit TR0
contrôle externe : le timer est lancé ou arrêté par transition négative (1 vers 0) sur INT0 si TR0 = 1

Timer 1 comme compteur

MODE	Fonction	VALEUR DE TMOD	
		contrôle interne	contrôle externe
0	TIMER 13 bits	40H	C0H
1	TIMER 16 bits	50H	D0H
2	RECHAR.AUTO 8 bits	60H	E0H
3	indisponible	-	-

Note :
contrôle interne : le timer est lancé ou arrêté par le bit TR1
contrôle externe : le timer est lancé ou arrêté par transition négative sur INT1 si TR1 = 1

Contrôle du TIMER 2 (sur 8052 seulement)

T2CON adresse directe C8H adressable au niveau bit

CFH	CEH	CDH	CCH	CBH	CAH	C9H	C8H
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

TF2: Dépassement de capacité. Mis à 1 lorsque le Timer 2 passe de la valeur FFFFH à la valeur 0.

EXF2: Indicateur externe. Mis à 1 lorsqu'une capture ou un rechargement est provoqué par transition 0 vers 1 sur broche T2EX et que EXEN2 = 1. Si l'interruption du Timer 2 est autorisée et que EXF2 = 1 alors le sous-programme placé à l'adresse 2BH est exécuté. La remise à 0 de EXF2 doit être assurée par programmation.

RCLK: Bit de contrôle de l'horloge de réception. RCLK = 1 positionné à 1, le port série, programmé en mode 1 ou 3, utilise l'impulsion de dépassement du Timer 2 comme signal d'horloge de réception. RCLK = 0, c'est l'impulsion de dépassement du Timer 1 qui joue ce rôle.

TCLK: Contrôle de l'horloge d'émission. TCLK = 1, le port série, (en mode 1 ou 3), utilise l'impulsion de dépassement du Timer 2 comme signal d'horloge d'émission. TCLK = 0, c'est l'impulsion de dépassement du Timer 1 qui joue ce rôle.

EXEN2: Validation du mode externe. EXEN2 = 1: autorisation de capture ou de rechargement provoqué par une transition négative détectée sur broche T2EX. Ceci n'est possible que si le Timer 2 n'est pas utilisé comme horloge du port série (TCLK et RCLK à 0). Le fait de forcer EXEN2 à 0 permet d'ignorer les événements présents sur la broche T2EX.

TR2 : Contrôle du fonctionnement. La mise à 1 de TR2 provoque le démarrage du Timer.

C/$\overline{T2}$: Sélection du mode de fonctionnement du Timer 2:
C/T2 = 0 fonction "temporisateur";
C/T2 = 1 fonction "compteur d'événements externes".
En mode temporisateur, c'est l'horloge (du microcontrôleur) divisée par 12 qui commande l'incrémentation du Timer. En mode compteur, seules des transitions négatives peuvent être détectées comme évènements externes.

CP/$\overline{RL2}$: sélection pour capture ou rechargement. CP/RL2 = 1 le mode capture est sélectionné. La conjugaison EXEN2 = 1 et détection d'une transition négative sur T2EX permet une opération de capture: la valeur contenue par le registre 16 bits du Timer 2 est placée dans les registres RCAP2H et RCAP2L. Ce mode de capture n'est possible que si RCLK et TCLK sont à 0. Dans le cas contraire, c'est le mode rechargement automatique qui est sélectionné.

INITIALISATION DU TIMER 2

exceptés les modes "générateur de fréquence de communication", ces procédures doivent être complétées d'un lancement du Timer par mise à 1 de TR2.

TIMER 2 en temporisation

MODE	VALEUR DE T2CON	
	contrôle interne	contrôle externe
Rechargement Auto. 16 bits	00 H (1)	08 H (2)
Capture 16 bits	01 H (1)	09 H (2)
Générateur de fréquence :		
pour émission et réception	34 H	36 H
pour émission seulement	24 H	26 H
pour réception seulement	14 H	16 H

TIMER 2 en compteur

MODE	VALEUR DE T2CON	
	contrôle interne	contrôle externe
Rechargement Auto. 16 bits	02H(1)	0AH(2)
Capture 16 bits	03H(1)	0BH(2)

Notes :
(1) Capture ou rechargement provoqué par dépassement de capacité du TIMER.

SCON : Contrôle du port série

Adresse directe 98H adressable au niveau du bit

9F	9E	9D	9C	9B	9A	99	98
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

mode de fonctionnement du port série :

SM0	SM1	Mode	description	fréquence de communication
0	0	0	registre à décalage	fosc./12
0	1	1	UART 8 bits	variable
1	0	2	UART 9 bits	fosc./64 ou fosc./32
1	1	3	UART 9 bits	variable

SM2 : Communication multiprocesseur possible en mode 2 et 3.
En mode 2 ou 3, si SM2 = 1, alors RI ne sera activé que si le 9^{e} bit est un 1.
En mode 1, si SM2 = 1, alors RI ne sera activé qu'à la réception d'un bit de stop valide.
En mode 0, SM2 doit être positionné à 0.

REN : Mis à 1 par programme REN autorise la réception série.

TB8 : correspond au 9^{e} bit à transmettre en mode 2 ou 3.

RB8 : correspond au 9^{e} bit reçu en mode 2 et 3. En mode 1, si SM2 = 0, RB8 correspond au bit se stop reçu.

TI : est l'indicateur d'émission. Il est automatiquement mis à 1 :
à la transmission du 8^{e} bit en mode 0 ;
à la transmission du bit de stop dans les autres modes.
Avant une opération d'émission, il doit être remis à 0 par programme.

RI : Indicateur de réception. Il est automatiquement mis à 1 :
à la réception de 8^{e} bit en mode 0,
à la réception du bit de stop dans les autres modes
(exception précisée dans la description de SM2).
RI doit être remis à 0 par programme.

Initialisation du port série

MODE	SCON	SM2
0	10H	
1	50H	SM2=0
2	90H	PROCESSEUR UNIQUE
3	D0H	
0	IMPOSSIBLE	SM2=1
1	70H	MULTIPROCESSEUR
2	B0H	
3	F0H	

Définition de la Fréquence de communication FC

Mode 0 :

$$FC = \frac{f_{osc}}{12}$$

Mode 1 :

FC est déterminée à partir du TIMER 1 configuré en mode "rechargement automatique"

$$TH1 = 256 - \frac{K \times f_{osc}}{384 \times FC}$$

où

K = si SMOD = 0
K = si SMOD = 1 (SMOD est un bit du registre PCON)

Mode 2 :

si

SMOD = 0 $\quad FC = \frac{f_{osc}}{64}$

SMOD = 1 $\quad FC = \frac{f_{osc}}{32}$

Mode 3 :

FC est variable. Même définition que pour le mode 1

Définition de FC à partir du TIMER 2 (8052 seulement) :

Le TIMER 2 est initialisé en mode "recharchement automatique". Il peut être commandé :

1 - par une horloge externe appliquée sur la broche T2

$$FC = \frac{\text{fréquence de dépassement de capacité}}{16}$$

2 - l'horloge externe f_{OSC}

$$FC = \frac{f_{OSC}}{32 \times [65536 - (RCAP2H,RCAP2L)]}$$

d'où

$$RCAP2H,RCAP2L = 65536 - \frac{f_{OSC}}{32 \times FC}$$

Utilisation des interruptions

SOURCE D'INTERRUPTION	INDICATEUR	ADRESSE DU VECTEUR
INT0	IE0	0003H
TIMER 0	TF0	000BH
INT1	IE1	0013H
TIMER 1	TF1	001BH
PORT SERIE	RI+TI	0023H
TIMER 2	TF2+EXF2	002BH

IE : Registre d'autorisation des interruptions

adressage au niveau du bit ADRESSE DIRECTE A8H

AF	AE	AD	AC	AB	AA	A9	A8
EA	-	ET2	ES	ET1	EX1	ET0	EX0

position	symbole	fonction
IE.7	AE	Inhibition de toutes les interruptions.
IE.6	—	sans fonction pour 8051 et 8052
IE.5	ET2	interruption TIMER 2
IE.4	ES	interruption PORT SERIE
IE.3	ET1	interruption TIMER 1
IE.2	EX1	interruption EXTERNE depuis broche INT1
IE.1	ET0	interruption TIMER 0
IE.0	EX0	interruption EXTERNE depuis broche INT0

ordre naturel de prise en compte des interruptions :

ordre	source	
1	IE0	plus haute priorité naturelle
2	TF0	
3	IE1	
4	TF1	
5	RI+TI	
6	TF2+EXF2	plus basse priorité

IP : Registre du niveau de priorité de l'interruption

adressage au niveau du bit adresse directe B8H

BF	BE	BD	BC	BB	BA	B9	B8
-	-	PT2	PS	PT1	PX1	PT0	PX0

position	symbole	fonction
IP.7	—	réservé à une fonction future
IP.6	—	réservé à une fonction future
IP.5	PT2	priorité interruption TIMER 2 (sur 8052)
IP.4	PS	priorité interruption port série
IP.3	PT1	priorité interruption TIMER 1
IP.2	PX1	priorité interruption externe INT1
IP.1	PT0	priorité interruption TIMER 0
IP.0	PX0	priorité interruption externe INT0

Le traitement d'une interruption ne peut être interrompu que par une autre interruption de niveau de priorité supérieur.

PCON : Registre de contrôle de la consommation

Adresse directe 87H

SMOD	—	—	—	GF1	GF0	PD	IDL

SMOD Mis à 1, ce bit assure un doublement de la fréquence de communication du port série obtenue à partir du TIMER 1.

GF1 Indicateur général sans fonction particulière. (1)
GF0 Indicateur général sans fonction particulière. (1)
PD Validation du mode Power-down. (1)
IDL Validation du mode IDLE. (1)

(1) : Ces bits n'existent que dans les versions réalisées en technologie CMOS

JEU D'INSTRUCTIONS DU 8051

Notes sur les symboles utilisés:

Rn:	Un des registres actifs de R0 à R7
direct :	Adresse d'un des 128 octets de RAM interne, ports, ou SFR.
@Ri :	Adressage indirect par registre R0 ou R1.
#data :	Donnée de 8 bits incluse dans le code exécutable.
# data16 :	Donnée de 16 bits incluse dans le code.
bit :	Adresse au niveau du bit d'un des 128 bits adressables.
rel :	Adresse relative au PC en complément à 2 de +127 à −128.
addr11 :	Une adresse limitée au bloc de 2Ko dans lequel figure l'instruction.
addr16 :	Une adresse sur l'espace de 64 Ko.

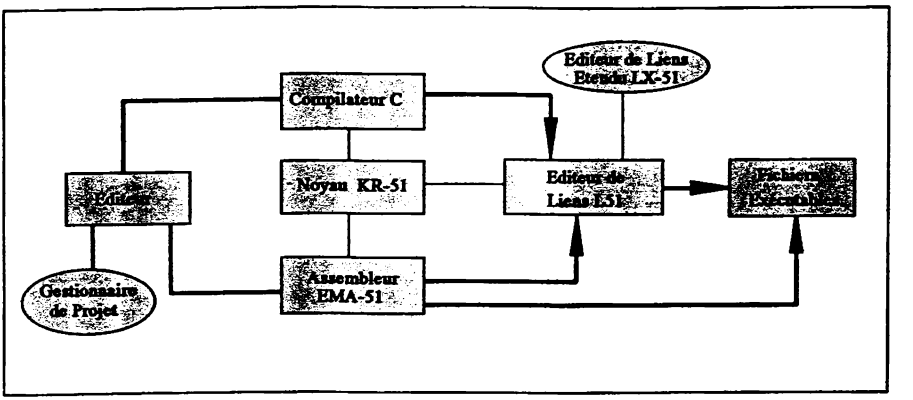
Mnémonique		Description	octets	cycle
OPERATIONS DE TRANSFERT				
MOV	A, Rn	Copie, dans A, la valeur du registre	1	1
MOV	A, direct	Copie, dans A, la valeur de l'octet adressé	2	1
MOV	A, @Ri	Copie, dans A, la valeur de l'octet adressé par Ri	1	1
MOV	A, #data	Place dans A la valeur de la donnée immédiate	2	1
MOV	Rn, A	Copie, dans le registre R, la valeur de A	1	1
MOV	Rn, direct	Copie, dans R, la valeur de l'octet adressé	2	2
MOV	Rn, #data	Place dans R, la valeur de la donnée	2	1
MOV	direct, A	Copie, dans l'octet adressé, la valeur de A	2	1
MOV	direct, Rn	Copie, dans l'octet adressé, la valeur de Rn	2	2
MOV	direct, direct	Copie, dans l'octet 1, la valeur de l'octet 2	3	2
MOV	direct, @Ri	Copie, dans l'octet adressé, la valeur adressée par Ri	2	2
MOV	direct, #data	Place dans l'octet adressé, la valeur de la donnée	3	2
MOV	@Ri, A	Copie, dans l'octet adressé par Ri, la valeur de A	1	1
MOV	@Ri, direct	Copie, dans l'octet adressé par Ri, la valeur d'un octet	2	2
MOV	@Ri, #data	Place dans l'octet adressé par Ri, la valeur de la donnée	2	1
MOV	DPTR, #data16	Place dans le registre DPTR, la donnée de 16 bits	3	2
MOVC	A,@A+DPTR	Place dans A, le code d'adresse (DPTR+A)	1	2
MOVC	A,@A+PC	Place dans A, le code d'adresse (PC+A)	1	2
MOVX	A,@Ri	Copie, dans A, la valeur de la RAM externe pointée par Ri	1	2
MOVX	A,@DPTR	Copie, dans A, la valeur de la RAM externe pointée par DPTR	1	2
MOVX	@Ri,A	Copie, dans la RAM externe adressée par Ri, la valeur de A	1	2
MOVX	@DPTR,A	Copie, dans la RAM externe adressée par DPTR, la valeur de A	1	2
PUSH	direct	Sauve dans pile, la valeur de l'octet adressé	2	2
POP	direct	Récupère depuis pile, une valeur placée dans l'octet adressé	2	2
XCH	A,Rn	Echange des contenus de A et de Rn	1	1
XCH	A,direct	Echange des contenus de A et de l'octet adressé	2	1
XCH	A,@Ri	Echange des contenus de A et de la RAM adressée par Ri	1	1
XCHD	A,@Ri	Echange des 4 bits de poids faible de A à RAM pointée par Ri	1	1
OPERATIONS ARITHMETIQUES				
ADD	A,Rn	A= A + Rn	1	1
ADD	A,direct	A= A + valeur de l'octet adressé	2	1
ADD	A,@Ri	A= A + valeur de l'octet adressé par Ri	1	1
ADD	A,#data	A= A + donnée immédiate	2	1
ADDC	A,Rn	A= A + Rn + retenue	1	1
ADDC	A,direct	A= A + valeur de l'octet + retenue	2	1
ADDC	A,@Ri	A= A + valeur de l'octet adressé par Ri + retenue	1	1
ADDC	A,#data	A= A + donnée + retenue	2	1
SUBB	A,Rn	A= A − (Rn + retenue)	1	1
SUBB	A,direct	A= A − (valeur octet + retenue)	2	1
SUBB	A,@Ri	A= A − (valeur octet adressé par Ri + retenue)	1	1
SUBB	A,#data	A= A − (donnée + retenue)	2	1
INC	A	A = A + 1	1	1
INC	Rn	Rn = Rn + 1	1	1
INC	direct	Ajoute 1 à la valeur de l'octet adressé	2	1
INC	@Ri	Ajoute 1 à la valeur de l'octet adressé par Ri	1	1
DEC	A	A = A − 1	1	1
DEC	Rn	Rn = Rn − 1	1	1
DEC	direct	Retranche 1 à la valeur de l'octet adressé	2	1
DEC	@Ri	Retranche 1 à la valeur de l'octet adressé par Ri	1	1
INC	DPTR	Ajoute 1 à la valeur 16 bits de DPTR	1	2
MUL	AB	Multiplication de A par B	1	4
DIV	AB	Division de A par B	1	4
DA	A	Ajustement décimal de A	1	1

JEU D'INSTRUCTIONS DU 8051 (suite)

Mnémonique		Description	octet	cycle
OPERATIONS LOGIQUES				
ANL	A,Rn	A = ET logique entre A et Rn	1	1
ANL	A,direct	A = ET logique entre A et octet	2	1
ANL	A,@Ri	A = ET logique entre A et valeur adressée par Ri	1	1
ANL	A,#data	A = ET logique entre A et donnée	2	1
ANL	direct,A	direct = ET logique entre A et octet adressé	2	1
ANL	direct,#data	direct = ET logique entre donnée et octet adressé	3	2
ORL	A,Rn	A = OU logique entre A et Rn	1	1
ORL	A,direct	A = OU logique entre A et octet	2	1
ORL	A,@Ri	A = OU logique entre A et valeur adressée par Ri	1	1
ORL	A,#data	A = OU logique entre A et donnée	2	1
ORL	direct,A	direct = OU logique entre A et octet adressé	2	1
ORL	direct,#data	direct = OU logique entre donnée et octet adressé	3	2
XRL	A,Rn	A = OU exclusif entre A et Rn	1	1
XRL	A,direct	A = OU exclusif entre A et octet	2	1
XRL	A,@Ri	A = OU exclusif entre A et valeur adressée par Ri	1	1
XRL	A,#data	A = OU exclusif entre A et donnée	2	1
XRL	direct,A	direct = OU exclusif entre A et octet adressé	2	1
XRL	direct,#data	direct = OU exclusif entre donnée et octet adressé	3	2
CLR	A	Mise à 0 de A	1	1
CPL	A	Complémente le contenu de A	1	1
RL	A	Rotation d'un bit, vers gauche, du contenu de A	1	1
RLC	A	Rotation d'un bit, vers gauche, du contenu de (A + retenue)	1	1
RR	A	Rotation d'un bit, vers droite, du contenu de A	1	1
RRC	A	Rotation d'un bit, vers droite, du contenu de (A + retenue)	1	1
SWAP	A	Echange des 4 bits poids fort avec 4 bits poids faible de A	1	1
OPERATIONS BOOLEENNES				
CLR	C	indicateur de retenue mis à 0	1	1
CLR	bit	bit mis à 0	2	1
SETB	C	indicateur de retenue mis à 1	1	1
SETB	bit	bit mis à 1	2	1
CPL	C	complémente l'indicateur de retenue	1	1
CPL	bit	complémente bit	2	1
ANL	C,bit	ET logique entre bit et retenue	2	2
ANL	C,/bit	ET entre complément du bit et C	2	2
ORL	C,bit	OU logique entre bit et C	2	2
ORL	C,/bit	OU entre complément du bit et C	2	2
MOV	C,bit	copie dans retenue C du bit	2	1
MOV	bit,C	copie de la retenue dans bit	2	2
SAUTS ET RUPTURES DE SEQUENCE				
AJMP	addr11	Saut absolu inconditionnel	2	2
LJMP	addr16	Saut long inconditionnel	3	2
SJMP	rel	Saut court par adressage relatif	2	2
JMP	@A+DPTR	Saut indirect à DPTR+A	1	2
JZ	rel	Saut si A = 0	2	2
JNZ	rel	Saut si A différent de 0	2	2
JC	rel	Saut si retenue à 1	2	2
JNC	rel	Saut si retenue à 0	2	2
JB	bit,rel	Saut si bit à 1	3	2
JNB	bit,rel	Saut si bit à 0	3	2
JBC	bit,rel	Saut si bit à 1 et bit=0	3	2
CJNE	A,direct,rel	Saut si A différent de l'octet	3	2
CJNE	A,#data,rel	Saut si A différent de la donnée	3	2
CJNE	Rn,#data,rel	Saut si registre différent de la donnée	3	2
CJNE	@Ri,#data,rel	Saut si octet pointé par Ri ‹› de la donnée	3	2
DJNZ	Rn,rel	décrémente registre et saut si résultat ‹› 0	2	2
DJNZ	direct,rel	décrémente octet et saut si résultat ‹› 0	3	2
GESTION DE SOUS-PROGRAMMES				
ACALL	addr11	Appel absolu de sous-programme	2	2
LCALL	addr16	Appel long de sous-programme	3	2
RET		Retour de sous-programme	1	2
RETI		Retour de sous-programme d'interruption	1	2
AUTRE				
NOP		Pas d'opération	1	1

All mnemonics copyrighted © Intel Corporation 1979

8051 Une gamme complète d'outils de développement



Outils de génération de code

	Type	Composants supportés	Remarques
SIMICE-51	Simulateur Universel	toute la famille	Simulation de tous les périphériques internes et de l'environnement externe.
EVA-51	Carte d'Evaluation	quasi totalité des composants (en version ROM externe)	Permet la réalisation rapide de prototypes.
MINI-31	Mini Emulateur pour le 80C31	80C31	Permet de bénéficier des fonctions d'Emulation les plus usuelles pour un investissement réduit.
[illegible]	Mini Emulateur Universel	quasi totalité des composants (en version Rom externe seulement)	Produit milieu de gamme.
[illegible]	Emulateur hautes Performances	quasi totalité des composants (y compris versions EPROM et ROM interne).	Emulation totalement transparente.

Outils de mise au point

RAISONANCE

ZI RUE DES SOURCES 38920 CROLLES FRANCE Tel : 76 08 18 16 Fax : 76 08 09 97

Imprimé par Jean-Lamour, 54320 Maxéville
Dépôt légal : mai 1996 - Dépôt légal de la 1re édition : 2eme trimestre 1993 *Imprimé en France*